NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A GRAPHICAL USER INTERFACE
FOR POST
(THE PROGRAM TO OPTIMIZE
SIMULATED TRAJECTORIES)

by

David Dean Nash

June 1995

Thesis Advisor: I. Michael Ross
Co-Advisor: Michael J. Zyda

Approved for public release; distribution is unlimited.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE June 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A GRAPHICAL USER INTERFACE FOR POST (THE PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES) (U)	5. FUNDING NUMBERS

6. AUTHOR(S)
Nash, David Dean

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The analysis and optimization of trajectories for aerospace vehicles has been extensively conducted by many government agencies using the Program to Optimize Simulated Trajectories (POST). The versatility of this program is made possible by its generalized planet and vehicle models, use of equality and inequality constraints, and multiple phase simulation capabilities. Unfortunately, it takes a "rocket scientist" to effectively use this program. For those who wish to have the power of this program without having to learn the required POST language, a Graphical User Interface (GUI) is necessary.

The GUI supports all the features of POST by offering the user selection windows that change depending on previous selections. An editable display window is the central portion of the GUI. As each selection is made from the event icons, the corresponding POST commands appear in the display window. This gives the experienced user the ability to switch between the new interface and the old file entry methods, and acquaints the new user with the POST file entry method. Once all selections are made the file can then be read by POST and the output used for analysis and visualization.

14. SUBJECT TERMS POST, GUI, interface	15. NUMBER OF PAGES 128
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

A GRAPHICAL USER INTERFACE FOR POST
(THE PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES)

David Dean Nash
Lieutenant, United States Navy
B.S., UCLA, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

June 1995

ABSTRACT

The analysis and optimization of trajectories for aerospace vehicles has been extensively conducted by many government agencies using the Program to Optimize Simulated Trajectories (POST). The versatility of this program is made possible by its generalized planet and vehicle models, use of equality and inequality constraints, and multiple phase simulation capabilities. Unfortunately, it takes a "rocket scientist" to effectively use this program. For those who wish to have the power of this program without having to learn the required POST language, a Graphical User Interface (GUI) is necessary.

The GUI supports all the features of POST by offering the user selection windows that change depending on previous selections. An editable display window is the central portion of the GUI. As each selection is made from the event icons, the corresponding POST commands appear in the display window. This gives the experienced user the ability to switch between the new interface and the old file entry methods, and acquaints the new user with the POST file entry method. Once all selections are made the file can then be read by POST and the output used for analysis and visualization.

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

The Program to Optimize Simulated Trajectories (POST) is "a general-purpose FORTRAN program for simulating and optimizing point mass trajectories of aerospace vehicles." [Ref 1] Martin Marietta Corporation modified the original Space Shuttle Trajectory Optimization Program for NASA to obtain the current version of 3D POST, Version 4.000. The capabilities of POST include targeting and optimization of point mass trajectories for powered and unpowered vehicles operating near a rotating, oblate planet. The program is quite generalized and allows for solving a wide variety of atmospheric flight mechanics and orbital transfer problems. POST was written in FORTRAN 77 for use on various platforms, including Silicon Graphics and Sun computers. Detailed information on the background, formulation and implementation of the POST program can be found in the three volume Final Report of The Program To Optimize Simulated Trajectories produced by the Martin Marietta Corporation. [Ref 1]

B. PROBLEM STATEMENT

To maximize the use of the current implementation of POST, it is necessary to have extensive knowledge of the system design language. A standard trajectory problem is defined as a sequence of at least two events that describe the initial conditions, environmental components, system constraints and desired output. The original version of POST required punch cards for this data entry, consequently data was organized in a proper sequence of Fortran phrases for each specific type of input, called *namelists*. As a hold over from this method, the data in the current version of POST must also be organized in a file in a proper sequence of *namelists*. The phrases for POST include $SEARCH, $GENDAT, $TBLMLT and $TAB. Each phrase is followed by Event variables and Hollerith input variables which must be entered in a specific format. As an example of the complexity of

1

an input file, an excerpt from Sample Problem 1 in the Utilization Manual [Ref 1] is provided in Figure 1. A complete version of this input file can be found in Appendix C.

```
$SEARCH
SRCHM      =              4,
IPRO       =              1,
MAXITR     =             10,
OPT        =            1.0,
OPTVAR     =  6HWEIGHT,
WOPT       =         1.0E-6,
CONEPS     =         89.98,
C
$GENDAT
EVENT      =              1,
NPC(2)     =              1,      4,      2,
NPC(8)     =              2,      1,
NPC(16)    =              1,
NPC(22)    =              1,
```

Figure 1: POST Excerpt

From the above excerpt it is apparent that, in its present state, POST is not an easy program to master. A significant amount of time is required to learn a new language and the relationship between all the variables. In order to make this powerful program available to more people, it was necessary to create a Graphical User Interface (GUI).

C. SCOPE

Development of a GUI for the POST program is the focus of this thesis. The purpose of the GUI is to make it easier for those not familiar with POST to use it without being concerned with all the intricacies of the program. At the same time, the GUI can be used to make targeting and optimization of trajectories more efficient for those familiar with the POST program. An option is provided to maintain the previous entry method for those not desiring to change. Default values are provided at all steps in the input process to facilitate the input procedure. Input values are checked wherever possible to ensure compliance with POST requirements. As an added feature, the ability to display the POST output file in a

2

3D mode on the Silicon Graphics machines has been added. The use of this graphics feature is briefly explained later in Chapter III, Section G.

There are numerous environments and methods available for developing user interfaces. Since the Sun and Silicon Graphics Incorporated (SGI) systems both use the X Window System running OSF/Motif, this was the environment chosen for the POST GUI (G-POST). There are also a number of tools available to prototype a graphic environment using X Windows and OSF/Motif. The primary design tool used for this project was the Builder Xcessory (BX) developed by Integrated Computer Solutions Incorporated (ICS) [Ref 2].

1. X Window System

X Windows was developed at the Massachusetts Institute of Technology (MIT) by personnel from MIT and Digital Equipment Corporation (DEC). It has been released in several versions, the most recent being Release 5 of Version 11 [Ref 3]. It is widely recognized as the standard for network-based windowing systems and is supported by a consortium of well-known companies such as AT&T, DEC, IBM, HP, Sun and SGI.

2. OSF/Motif

Motif is a GUI built on top of X Windows. The developer of Motif, the Open Software Foundation (OSF), is a non-profit organization founded in 1988 by a group of companies (including DEC, HP and IBM) whose objective was, and is, to develop a standard user-interface environment [Ref 3]. Motif uses a set of components called widgets to provide the GUI applications with the capability of running on practically any vendor's workstation.

3. The Builder Xcessory (BX)

The BX has a palette of the most commonly used widget shells (the windows that hold all the buttons), widgets (buttons, sliders, selectors, lists etc.) and composite widgets (more complicated widgets shells with other widgets already attached). The OSF/Motif standard as described above uses widget shells of various types as containers for all other widgets.

BX follows these standards by allowing the user to drag copies of the desired widgets over to the work area and define their positions and characteristics graphically. Each type of input widget; push-button, toggle-button, slider, and text entry areas, has procedures associated with them termed callbacks. These callbacks register a change in the state of the widget (e.g., a Help button pushed), and execute the appropriate action (e.g., open the Help window). Once all shells are constructed to the user's satisfaction, a play mode may then be selected to view the window connections and make sure, when the proper button is pressed, the program operates as expected [Ref 3].

When the user is satisfied with the general operations of the window environment, the Builder Xcessory can then generate the code for the GUI in C, C++ or the Motif User Interface Language. The BX uses its own code along with the OSF/Motif code to accomplish all windowing functions. The non-window code is then added to the windowing environment to allow the program to perform its desired functions.

D. RESEARCH APPROACH

Initial research began with a comprehensive study of the methodology POST uses to accomplish its many tasks. An excellent overview of POST, *A Primer for POST* written by John Nicholson in 1993 [Ref 4], was used as a guide in determining the direction of research. A GUI is usually integrated as part of a program, written in the same language and started with the program. In this case, after review of Nicholson's Primer and the POST manuals, it was determined that it would be most effective to write an interface as part of a file generator that could then be read by POST to obtain the desired output.

To develop this file generator, it was necessary to fully understand the relationship of all POST commands. All variables were mapped out and cross referenced with their requirements and the variables they affected. Appendix A shows a summary of POST commands and related functions. A complete listing of all features can be found in the POST users manuals [Ref 1]. Only after extensive study of POST could the requirements for the GUI be determined. The interrelationship between all data entries also increases the

complexity of the windowing system used for the GUI. As an example, each option that is no longer available due to a previous selection must be made inaccessible, or "grayed-out". Once the connections were mapped out, the windows for the interface could be designed.

All windows for this thesis were initially constructed using the BX toolkit, which makes the design and positioning of the widgets in the shell relatively fast once the program is mastered. C code was generated using BX to provide an outline for the actual code used in the interface. This was done to insure compatibility with previously written code which contained routines to decrease the amount of code required and make the program more modular. These routines will be discussed in Chapter III, Section B.

E. ORGANIZATION

An in-depth description of the architectural design and interface requirements for POST is detailed in Chapter II. These requirements are then translated into a user interface design in Chapter III. The actual interface implementation is shown with a tutorial in Chapter IV. Conclusions and recommendations are found in Chapter V.

II. INTERFACE REQUIREMENTS AND ARCHITECTURAL DESIGN

A. INTERFACE REQUIREMENTS

In order to function as an effective tool for the new user, the interface must not only contain all possible data entry fields, but also supply those entry fields in a logical and systematic order. The primary goal of the GUI is to produce an input file which conforms with the structured entry format required by POST, while making the data entry format intuitive. This is especially important since the method one user formulates a problem statement with might be different from the way another does, and neither formulation may be the order in which POST accepts the data entry. For instance, POST requires targeting and optimization entries before any initial vehicle conditions such as position and velocity are entered. It is more logical to most when formulating the trajectory problem to begin with initialization conditions prior to targeting and optimization parameters. With this in mind, a step-by-step entry method is implemented to ensure all necessary data fields are recorded in the proper order.

Complete file operations and editing functions are made available for the extensive amount of file manipulations required when creating a POST input file. Help functions are also accessible at each step, displaying the information found in the manual on the current subject, as well as any interface specific aspects of that window. Default values of all variables, as described in the reference manuals, are set until user modified, at which time these values are stored until the program is exited and restarted. Wherever possible, requesters ask for confirmation of actions that might adversely affect the current input file. The Motif style guide is followed to ensure consistent behavior with the OSF/Motif user interface framework. The interface runs on Silicon Graphics workstations and is written such that recompiling on Sun workstation is relatively simple.

B. ARCHITECTURAL DESIGN

There are four major functions available in G-POST: create a new or edit an existing POST input file using the Event Worksheet, generate a POST output file from a previously constructed input file, analyze the output data, and run a visual 3D representation of the trajectory. These four functions are modules that are accessed individually from the G-POST initial selector (Figure 2) and are used to manipulate POST input or output files. An

Figure 2: The G-POST Initial Selector

execution of the POST program consists of the following steps as described on page 3.a-1 of the POST Utilization Manual [Ref 1]:

- Initializing the equations of motion.
- Propagating the trajectory until interrupted by the occurrence of the user-specified conditions for the next event.
- Reinitializing the equations of motion with new user inputs for the event causing the interruption.

- Repeating the process until the user-specified final event is reached.
- Terminating the problem.

The POST trajectory problem is first defined as a sequence of at least two events. These events are constructed in the proper order using the Event Worksheet.

1. Event Worksheet

As the creation tool for the POST input file in G-POST, the Event Worksheet is the interface between the user and the input file. User input is accomplished either through numerous Motif widgets designed to create the input file in the most intuitive manner possible, or direct input entry into the multi-line text entry widget (Figure 3). Further modifications of the POST input file can be done using the file editing functions from the pull-down menus. The program can only be terminated from the initial shell, all other close functions unmap the individual shell. If a closed shell is requested an additional time, the shell is remapped and not recreated. This allows for retention of values previously entered for that shell. The Event Worksheet contains button widgets which are used to bring up all of the other widget shells that are required to construct the POST input file. When all modifications are complete, the input file can be saved for future manipulations or to create a POST output file.

2. Run POST File

Any previously created POST input can be submitted to the POST program using G-POST as long as the POST program is located on the machine where G-POST is located, or can be reached via a link or alias. If any *namelist* errors or unexpected results occur during job submission, the errors can be corrected and the problem formulation reviewed with the Event Worksheet and resubmitted. The output file can now be analyzed for any desired output.
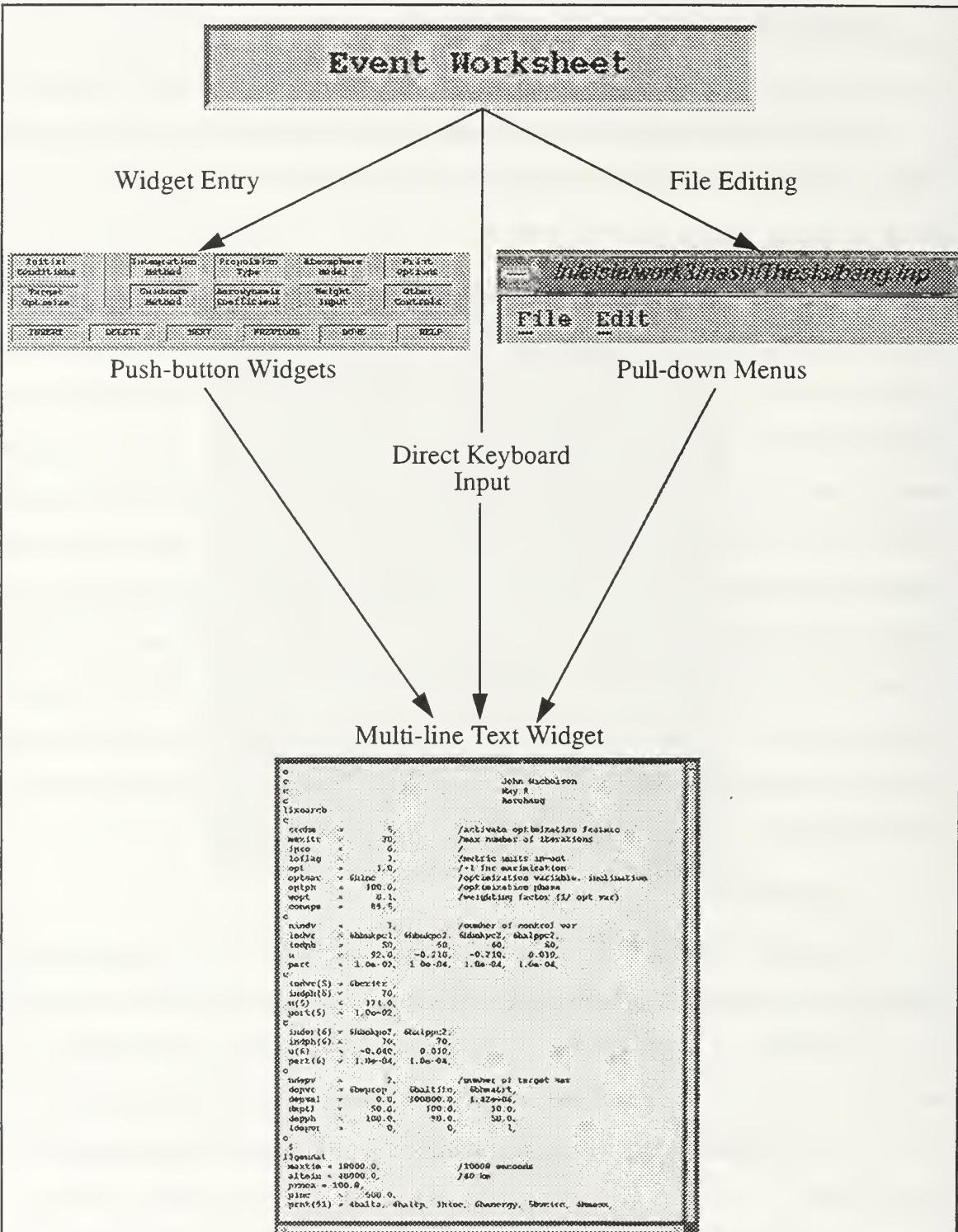
Figure 3: The Event Worksheet Input Methods

3. Analyze Data

The standard output file from the POST program is in a form that is not easily read or used for problem analysis. A *Profil* file generation option is available to create either an ASCII or binary file that can more readily be used by a number of programs to analyze the data. The current implementation is to convert the *Profil* file to a Matlab [Ref 5] file which allows for graphing of any of the output variables for data analysis. The method of accomplishing this is discussed later in Chapter III, Section F.

4. Run Simulation

The simulation program is started from G-POST, but runs independently from G-POST and takes a POST *Profil* as input. The present implementation requires inertial pitch, roll, yaw, inertial x, y and z-position, and time. These values are used to generate the simulation using an OpenGL application running on the SGI workstation.

C. WIDGET DESIGN

When describing any of the specific GUI widgets in the following sections, *italics* are used with the widget name as it appears in the GUI (e.g., *Event Worksheet*). The name of a container shell is in normal Title Case (e.g., Row Column) and any generic children names are left in lower case (e.g., push-button).

1. Shell Widgets

The first window to appear when the program is started is called the application window. This window is defined using an ApplicationShell. Only one of these shells is allowed in each program. To define all other shells in the program a TopLevelShell structure was used since all windows defined after the initial application window are at the same level in the widget hierarchy. A TopLevelShell is a child of the ApplicationShell and inherits all the windows' resources available from the ApplicationShell. Closing or destroying any of the TopLevelShells does not quit the program as closing the ApplicationShell does.

2. Container Widgets

Two types of container widgets, Form and Bulletin Board, are defined as children of the TopLevelShell. A Form widget is a container that allows for the resizing of its children if desired. The children of the Form are attached to its sides and each other to maintain their relationship when the Form is resized. Resize capabilities are not used with any of the shells in this interface and the actual resize option was not implemented. Minimization of any of the shells to icons is still allowed.

A Bulletin Board widget is a container that does not resize its children and requires specification of all children positions instead of attachments. This allows for more control of widget positioning, but is more difficult than using Form widgets which automatically position its children. The use of BX made this procedure easier by defining the positions when the children were placed in the Bulletin Board. Moving any of the Bulletin Board's children after generation of code was a matter of specifying the new coordinates of the widget.

Children of these two containers included two more containers, the Row Column and Scrolled Window widgets, and all the labels, separators, text fields and push-buttons not located in the Row Column and Scrolled Window containers. The Row Column container orders and groups its children; push-buttons, toggle-buttons and text fields, for organization and referencing by a single name. Toggle-buttons can be put in a Radio Box version of the Row Column container to force the toggle-buttons to behave in a "One-of-many" manner (i.e., only one button can be pressed at a time).

Multi-line text editing and selection boxes are placed in the Scrolled Windows to allow access to the hidden data in the window. These are operated using the arrows and sliders located on the right side (and bottom if necessary) of the container.

III. USER INTERFACE DESIGN

A. INTERFACE PRINCIPLES

As stated in the requirements section, G-POST follows the OSF/Motif style guide in the creation of all the widgets in the interface. Specifically:

- Know the User.
- Empower the user.
- Keep interfaces flexible.
- Use progressive disclosure.
- Allow direct manipulation.
- Provide rapid response.
- Make navigation easy.
- Keep interfaces consistent.
- Use explicit destruction.

The following sections of this chapter demonstrate these principles by showing the actual window as displayed to the user and describing the underlying design.

B. GENERIC ROUTINES

As mentioned in the introduction, routines were developed to reduce the amount of code necessary to create all widgets as well as facilitate the conversion to a C++ class structure at a later date. Since the structure of an individual widget type is relatively the same from one instance to the other, a procedure can be developed to produce these widgets so that all that is required is one call to the procedure for each widget. The creation of labels, separators, push-buttons, toggle-buttons and text fields make up the majority of the interface design. Up to about 400 text field entries could be required in a given trajectory problem. As many labels as text fields and nearly that many buttons are also required. The Motif definition of these "simple" widgets can require anywhere from seven to fifteen lines

of code for each widget. Without these routines the length of the program would be unmanageable. Figure 4 shows the make_textfield_entry routine which is used to make all the text field widgets in the GUI.

```
void make_textfield_entry(
    Widget &_textField,
    Widget rc,             // A row column container.
    char *entrytext,       // Text for the label string.
    int text_columns,
    char * text_value,
    int text_width,        // Width of textfield
    int text_height,       // Height of textfield.
    XtCallbackProc callback, // Name of procedure to
            // call when this toggle button is selected.

    XtPointer user_data)   // Data to be sent the
                           // callback.
{
    Arg wargs[10];    // Same old args stuff.
    int n;            // Same old args stuff.

    // Create this particular textfield.
    n = 0;
    XtSetArg(wargs[n], XmNhighlightOnEnter, True); n++;
    XtSetArg(wargs[n], XmNcolumns, text_columns); n++;
    XtSetArg(wargs[n], XmNwidth, text_width); n++;
    XtSetArg(wargs[n], XmNheight, text_height); n++;
    XtSetArg(wargs[n], XmNvalue, text_value); n++;
    _textField = (Widget) XmCreateTextField(rc, entrytext,
                    wargs, n);
    XtManageChild(_textField);
    XtAddCallback (_textField, XmNvalueChangedCallback,
                    callback, user_data);
}
```

Figure 4: Text Field Entry Routine

The procedure takes a reference to the text field widget name so that data can be passed into and read from any individual global text field. The data in the text fields must be accessed and manipulated individually vice in whole sections as is the case with buttons. The desired widget characteristics are passed to the appropriate calls and the widget is managed.

Modifications on the code originally used in *rotate3* [Ref 6] were used for the initial design of the toggle-button routine. The other routines use the same format as the text field example except their individual widget names are not important. Toggle-buttons, labels and separators are initialized as gadgets instead of widgets. Gadgets are widgets without their own windows, they become part of the parent rather than a child of the parent. This saves resources and speeds up operation. Push-buttons are more complex, requiring the use of widgets for those functions that bring up message windows (e.g., Help button and file requesters) when activated.

C. THE G-POST SHELL

When the program is begun (by typing gpost), the initial shell, *G-POST* (Figure 5), is displayed with the four primary options as described in the architecture section of Chapter II, as well as one button to *EXIT* the program and another to *HELP* in getting started. The
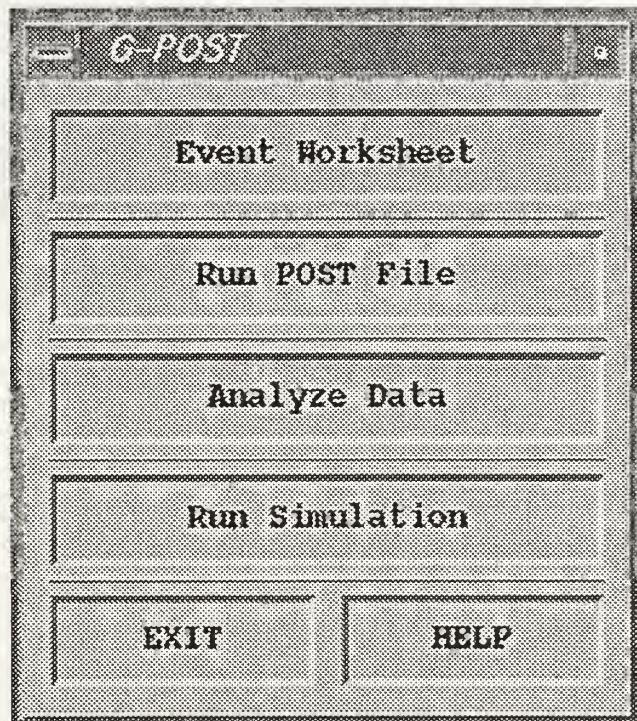
Figure 5: The Initial Shell

G-POST shell is created using a Form widget with push-button widgets used for each option. Selection of any of the button widgets makes the appropriate callback to start the desired module.

The *Event Worksheet* push-button brings up a file requester consisting of a Bulletin Board Dialog shell with two buttons (Figure 6). A Dialog shell is a shell that appears to give a message or request an input, and then goes away once the selection is made. When pressed, the *Create NEW File* button displays a prompt dialog which takes as input a new file name (Figure 7).
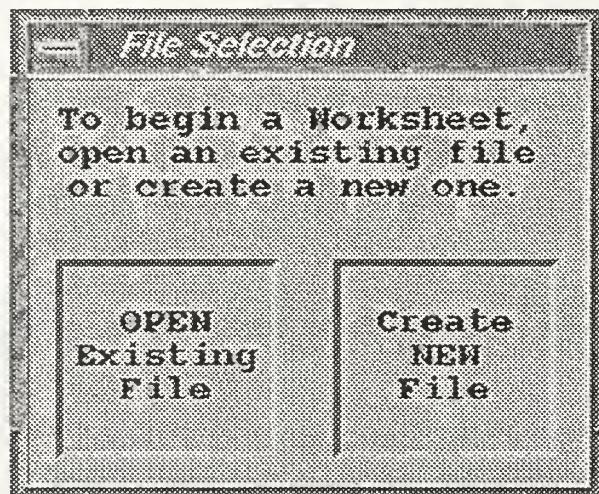
Figure 6: The File Selection Shell

16
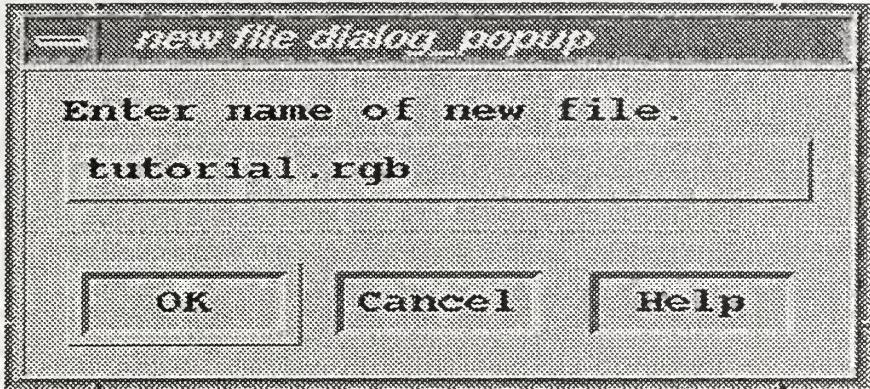
Figure 7: The New File Dialog Shell

The *OPEN Existing File* button calls a file selection dialog popup (Figure 8) with the file filter set for the POST input file extension (*.inp). Once an input is made in either dialog popup, the Event Worksheet main shell is opened.
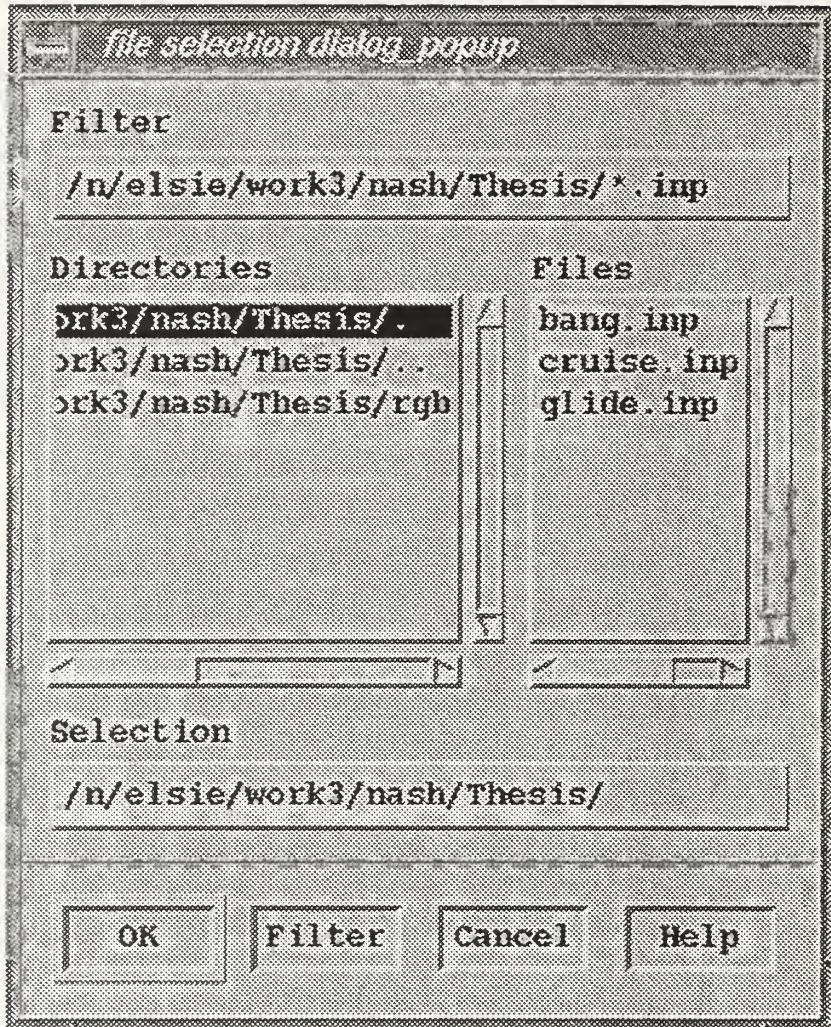
Figure 8: The Open File Selection Shell

D. THE EVENT WORKSHEET

The *Event Worksheet* main shell (Figure 9) is created using a TopLevelShell and a Bulletin Board container. Two function calls, one for the menu bar and one for the push-buttons and multi-line text entry widget, are made to create all the children of the Bulletin Board. The menu bar is attached to the Bulletin Board and contains three pull down menu widgets with the required buttons for *File* manipulation (*Open, New, Close, Save, Save As, Print* and *Exit*), *Edit* (*Cut, Copy, Paste and Clear*) and *Help* (*Help* and *Program Version*)

18

functions. Keyboard shortcuts are associated with each of the buttons under _File_ and _Edit_ by typing the underlined letter.

The body of the of the Bulletin Board consists of three Row Column containers; one for the _Initial Conditions_ and _Target Optimize_ buttons, one for the _INSERT_, _DELETE_, _NEXT_ and _PREVIOUS_ buttons, and one for all the other buttons except _DONE_ and _HELP_. Three separators and the multi-line text entry widget are also defined.

Button widgets on the worksheet are active or inactive depending on whether the initial selection was to open a new file or edit an existing one. A new file requires initial conditions before targeting information and event criteria are entered. All buttons, except the _Initial Conditions_ button, will be realized with their sensitivity resource set to False (i.e., grayed out). An initial selection of _OPEN Existing File_ will cause the _Initial Conditions_ and _Target Optimize_ buttons to be realized with their sensitivity resource set to False, and all the other buttons to be set active. All file and edit features are available in either case, allowing for a new or existing file to be opened in place of the original selection. Construction of a new POST input file begins with the selection of _Initial Conditions_, and continues with _Target Optimize_ once initialization is complete. The event selections are available after targeting and optimization is completed.

The event selection buttons (_Integration Method, Guidance Method, Propulsion Type, Aerodynamic Coefficients, Atmosphere Model, Weight Input, Print Options_ and _Other Controls_) are available if _OPEN Existing File_ was selected, or if initialization, targeting and optimization were completed after selecting _Create NEW File_. These buttons, along with the event control parameters buttons (_INSERT, DELETE, NEXT_ and _PREVIOUS_), allow for entry of new criteria at each successive event. The _DONE_ button will close the _Event Worksheet_ shell if the current file has been saved, or it will call a prompt dialog to determine if the action requested is truly desired. The _HELP_ push-button describes the operation of the other push-buttons, whereas the menu _Help_ describes the file and edit features. The operation of all push-buttons will be described in the following sections.

Initial Conditions | Integration Method | Propulsion Type | Atmosphere Model | Print Options

Target Optimize | Guidance Method | Aerodynamic Coefficient | Weight Input | Other Controls

INSERT | DELETE | NEXT | PREVIOUS | DONE | HELP

```
c
c                                        John Nicholson
c                                        May 8
c                                        Aerobang
l$search
c
  srchm    =        5,         /activate optimization feature
  maxitr   =       30,         /max number of iterations
  ipro     =        0,         /
  ioflag   =        3,         /metric units in-out
  opt      =      1.0,         /+1 for maximization
  optvar   = 6hinc   ,         /optimization variable, inclination
  optph    =    100.0,         /optimization phase
  wopt     =      0.1,         /weighting factor (1/ opt var)
  coneps   =     89.5,
c
  nindv    =        7,         /number of control var
  indvr    = 6hbnkpc1, 6hbnkpc2, 6hbnkpc2, 6halppc2,
  indph    =       50,       50,       60,       60,
  u        =     92.0,   -0.210,   -0.210,    0.010,
  pert     = 1.0e-02, 1.0e-04,  1.0e-04,  1.0e-04,
c
  indvr(5) = 6hcritr ,
  indph(5) =       70,
  u(5)     =    171.0,
  pert(5)  = 1.0e-02,
c
  indvr(6) = 6hbnkpc2, 6halppc2,
  indph(6) =       70,       70,
  u(6)     =   -0.040,    0.010,
  pert(6)  = 1.0e-04,  1.0e-04,
c
  ndepv    =        2,         /number of target var
  depvr    = 6hwprop , 6haltito, 6hheatrt,
  depval   =      0.0, 300000.0,  1.42e+06,
  deptl    =     50.0,    100.0,     10.0,
  depph    =    100.0,     90.0,     50.0,
  idepvr   =        0,        0,        1,
c
  $
l$gendat
  maxtim = 10000.0,             /10000 seconds
  altmin = 40000.0,             /40 km
  prnca = 100.0,
  pinc   =     500.0,
  prnt(91) = 4halta, 4haltp, 3hinc, 6henergy, 5hvcirc, 4hmass,
```

Figure 9: The Event Worksheet Shell

20

1. The Initial Conditions Shell

When opening a new file or selecting *Initial Conditions* from the *Event Worksheet*, the *Initial Conditions* shell (Figure 10) is displayed. All required entries for the first event under $GENDAT are found here. The shell is constructed using a Bulletin Board widget holding the nine push-button widgets, separator gadgets and three push-buttons which are standard on all the shells in the interface: *OK*, *CANCEL* and *HELP*.
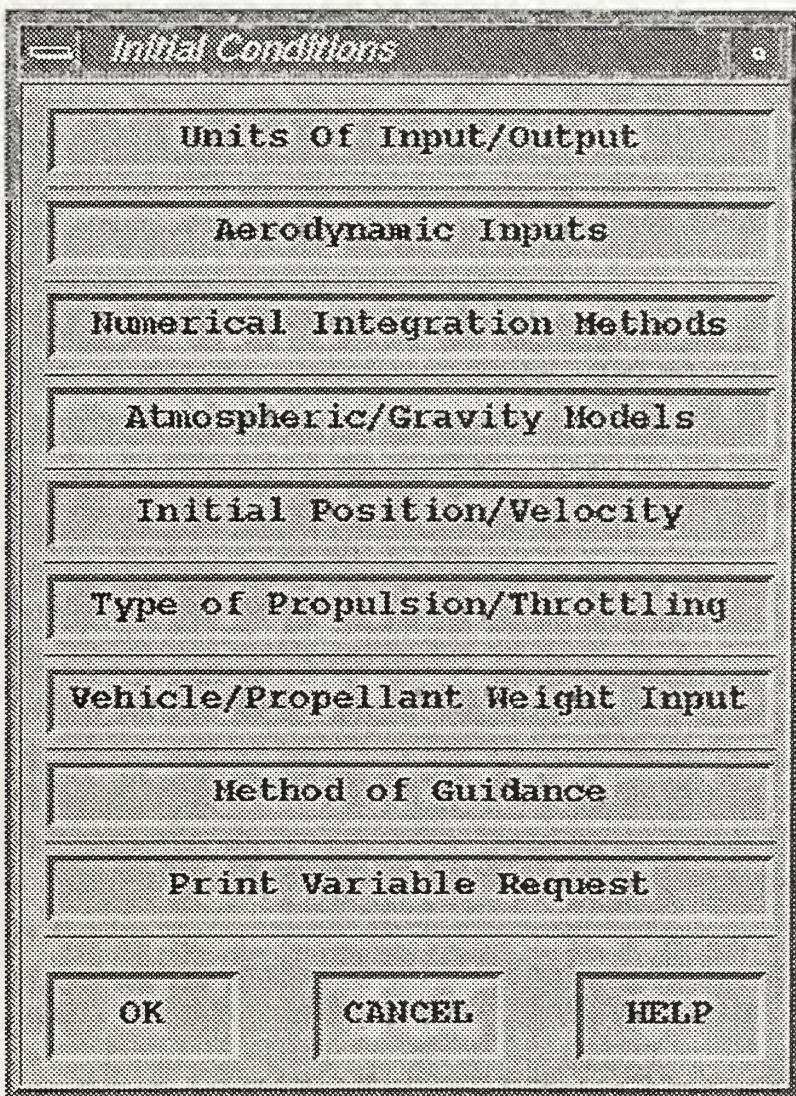
Figure 10: The Initial Conditions Shell

Each of the nine initial conditions buttons, beginning with *Units of Input/Output,* make calls to construct at least one other widget shell, depending on which of the initial conditions is pressed. Selection of *OK* will print the input values as modified by the nine selectors to the multi-line text widget on the *Event Worksheet.* If any fields are not modified, the default values stored in the initial conditions shells are printed to the text widget. *CANCEL* exits the window without printing any values and *HELP* provides information on all the buttons.

2.   Units Of Input/Output

POST can take English or metric units as input and print the output using English or metric units. All input units must be of the same type and all output units are also of the same type. If a conversion is necessary, POST uses the modifiable stored values for each conversion factor. The widget (Figure 11) consists of four toggle-buttons (*English/English, English/Metric, Metric/English* and *Metric/Metric*), and the standard *OK, CANCEL* and *HELP* push-buttons.
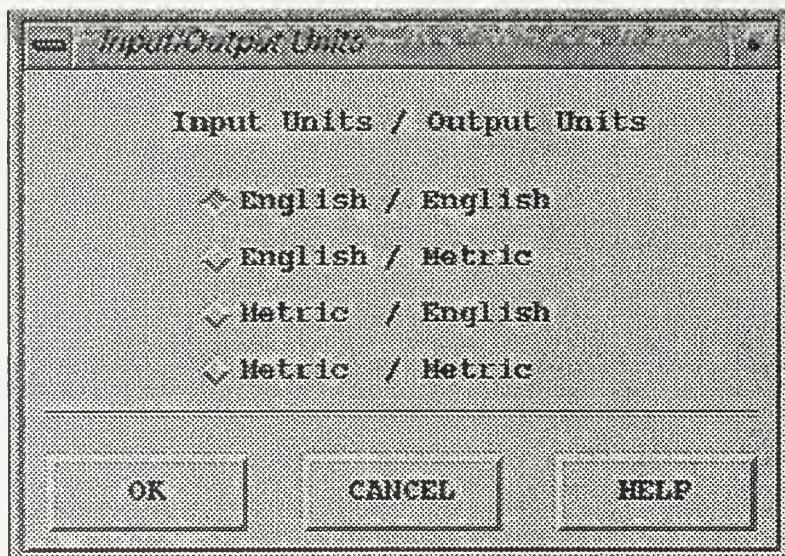
Figure 11: The Units Shell

22

3.   Aerodynamic Inputs

The *Aerodynamic Inputs* shell (Figure 12) consists of a Bulletin Board shell holding Row Column and Radio Box containers, text field, toggle-button, label and separator gadgets, and the three standard push-buttons. There are five selections available for defining the aerodynamic characteristics of the vehicle. The aerodynamic coefficients are input as tables by selecting the desired toggle-button widget. Any call in the program to produce a table brings up an open file selection requester (like Figure 8) that looks for files with the .tab ending. A reference to this file's contents is stored for later printing in the table section ($TAB) of the *Event Worksheet* when the *OK* button is pressed. Selecting *CANCEL* exits with no changes to the *Aerodynamic Inputs* shell.

Viscous Aerodynamic Coefficients selection requires input of additional data. Text fields are available for entry of viscous components used with the viscous aerodynamic coefficients tables. The default values are displayed and grayed out whenever any selection other than *Viscous Aerodynamic Coefficients* are selected.

Selection of *Static Trim Aerodynamic Coefficients* displays the *Static Trim* shell (Figure 13). This window also consists of a Bulletin Board shell containing Row Column containers, toggle-buttons, text field widgets and the same three push-buttons. A selection in the *Static Trim Option* area other than *None* must be selected before any other selections or data entry can be made.
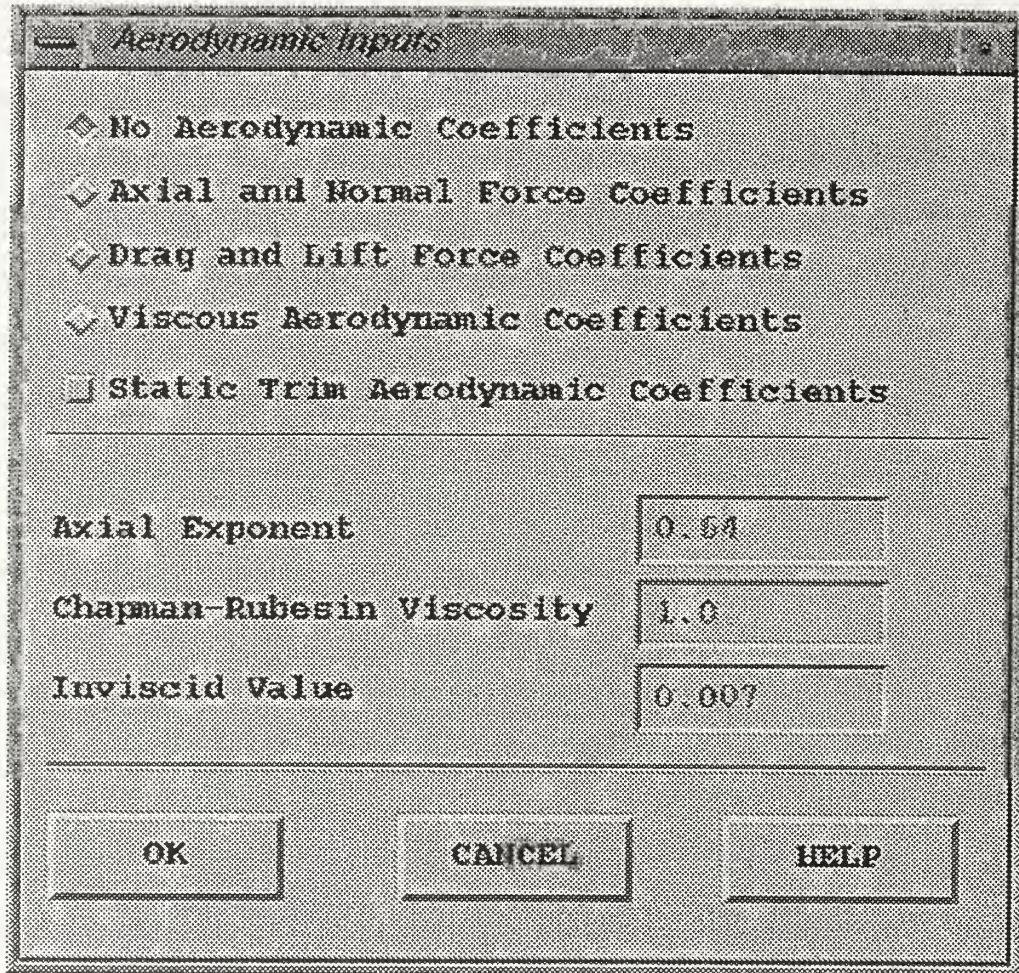
Figure 12: The Aerodynamic Inputs Shell

Figure 13: The Static Trim Shell

4. Numerical Integration Methods

The third choice on the *Initial Conditions* window brings up the *Methods of Integration* shell (Figure 14). This is the first of many complicated shells. This shell consists of multiple Radio Box and Row Column containers, with toggle-buttons, separators, labels, text field widgets, and the three standard push-buttons. The Radio Boxes separate the selections into three categories: *Integration Methods*, *Special Integration Step Size* and *Conic Calculations*. The default values are set in each text field and on each toggle-button. In order to enter values for *Max Step Size*, *Min Step Size*, *Number of Steps* and *Max # Steps*, *Krogh Predictor-Corrector* must be chosen from the *Integration Methods* section. The *Increment in True Anomaly* is available for selection when *Calculation At End Of Integration Step* in the *Conic Calculations* is selected. If this option is selected the *Step Size* input is no longer used and the text field is grayed.
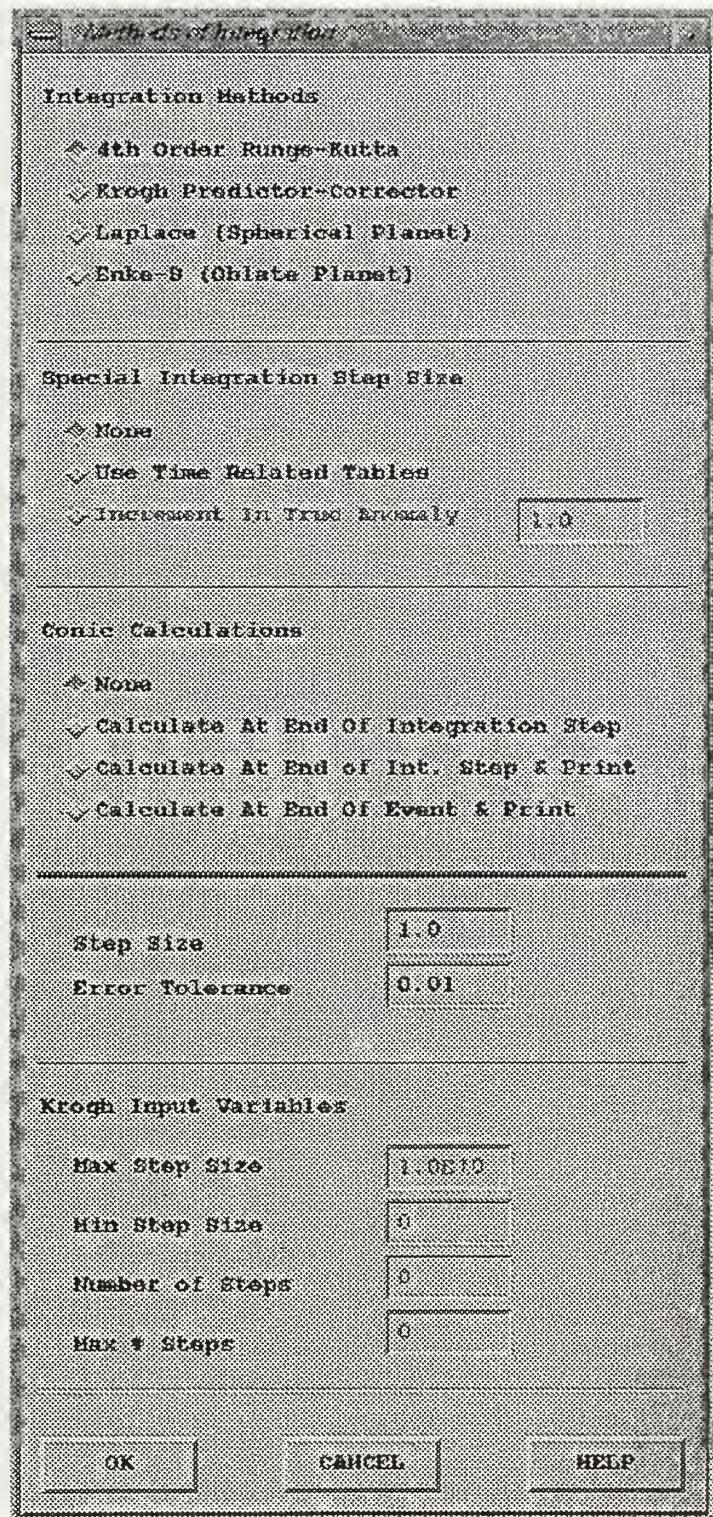
Figure 14: The Integration Method Shell

5. Atmospheric/Gravity Models

The modeling methods for the atmosphere and gravity are chosen in the *Atmospheric/ Gravity Models* shell (Figure 15) using a Bulletin Board shell to hold Radio Box and Row Column containers, toggle-button and text field gadgets, and the standard three push-buttons. Default widget selections include: *1962 US Standard* atmosphere model, no *Winds*, no *Aeroheating* or *Special Aeroheating*, and the *Oblate Planet* gravity model.

Gravity Coefficients (harmonics in the gravity potential function - *J2* through *J8*, equatorial radius - *RE*, polar radius - *RP*, *MU* and *Omega*) can all be modified if desired when *Oblate Planet* is selected in the *Gravity Model* section. If *Spherical Planet* is chosen only *RE* and *MU* are available for modification.

POST has stored values for various standard atmospheres and can model any atmosphere that can be described by tables of atmospheric density, pressure, temperature and speed of sound. Choosing any model other than *None* allows for selection of *Winds, AeroHeating* and *Special AeroHeating*. *Winds* can be defined using tables of speed and azimuth or as northerly and easterly values. Either case will allow input of *Turbulence* and/ or *Gusts*. If *Gusts* are chosen the *Frequency* and *Amplitude* text fields become available for entry.

Atmosphere and Gravity Models

Atmosphere Model
- Tables
- 1962 US Standard
- 1963 Patrick AFB
- 1971 Vandenberg AFB
- 1976 US Standard
- Random
- None

Winds
- None
- Speed/Azimuth Tables
- Northerly/Easterly Tables
- Turbulence
- Gusts

Aeroheating
- None
- Chapman
- Table Lookup
- Chapman & Table Lookup
- Turbulent Heating Only
- Laminar & Turbulent Heating
- Max Centerline Heat Rate

Special Aeroheating
- None
- 10-Panel Vehicle
- All Special Indicators
- Stagnation Point Only
- Cylindrical

Gravity Model
- Oblate Planet
- Spherical Planet

Table Constants

CS	0
DENS	0

Gusts
Frequency - deg/sec

X	0
Y	0
Z	0

Amplitude - ft/s(m/s)

X	0
Y	0
Z	0

Gravity Coefficients

J2	1.0826271E-3
J3	-2.5358866E-6
J4	-1.6246180E-6
J5	-0.22698599E-6
J6	0.5418572E-6
J7	-0.36319255E-6
J8	-0.20772554E-6
RE -ft	20925741
RP -ft	20855590
MU -ft3/s2	1.4076539E16
OMEGA-rad/s	7.29211E-6

OK	CANCEL	HELP

Figure 15: The Atmosphere And Gravity Model Shell

Aeroheating is calculated using Tables (via file selector), *Chapman* values, a combination of the two (for laminar or turbulent flow), or using *Max Centerline Heat Rate*. If any of the *Chapman* options are selected, another shell is displayed (Figure 16) and the *10-Panel Vehicle* option under *Special Aeroheating* is enabled. The *Chapman* shell consists
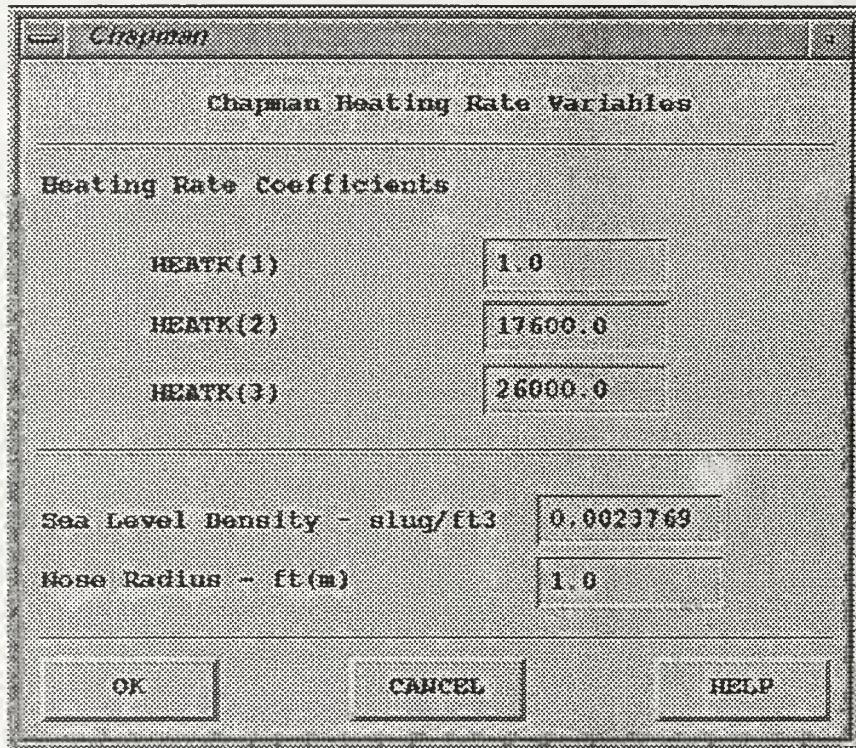
Figure 16: The Chapman Coefficients Shell

of a Bulletin Board holding Row Column containers, text fields, labels and separators, and the standard push-buttons. Text field entries are provided in the *Chapman* shell for the required *Chapman* variables. The default values for the Chapman coefficients are displayed and can be altered if desired.

Special Aeroheating values can be calculated depending on the toggle-button selection in the *Special Aeroheating* section. Selection of *10-Panel Vehicle* displays an additional shell (Figure 17) consisting of a Bulletin Board shell, two Row Column containers, each with fifteen text fields, labels, separators and the standard three push-

buttons. The text fields allow for input of panel *Surface Area*, *Heat Ratio* and the *Weight/Area* table to use with the *10-Panel Heating Calculations*.

Figure 17: The 10-Panel Vehicle Shell

6. Initial Position/Velocity

Position and velocity can be entered in various coordinate systems depending on toggle-button selection. The *Position And Velocity Input* shell (Figure 18) provides Radio Box and Row Column containers, toggle-buttons, text fields, labels and three standard push-buttons for this entry. The default selection is *Spherical Position/Earth Relative Velocity* with zero values for all vectors. Default representation of the *Spherical Position* uses *Altitude*, *Geodetic Latitude* and *Relative Longitude* toggle-button selection.

31

Initial Position And Velocity

◇ Earth Centered Inertial Position And Velocity

◇ Spherical Position/Local Inertial Velocity

◇ Spherical Position/Atmosphere Relative Velocity

◆ Spherical Position/Earth Relative Velocity

◇ Orbital Parameters

Spherical Position

◆ Altitude   ◆ Geodetic Latitude   ◆ Relative Longitude

◇ Geocentric Radius   ◇ Geocentric Latitude   ◇ Inertial Longitude

| 0 | | 0 | | 0 |

Orbital Parameters		ECI Position Vector	
		X - ft(m)	0
Perigee Altitude - NM(KM)	0	Y - ft(m)	0
Apogee Altitude - NM(KM)	0	Z - ft(m)	0
Inclination - DEG	0		
Long. Ascending Node - DEG	0	ALL Velocity Vectors	
Argument of Perigee - DEG	0	X - ft/s(m/s)	0
True Anomaly - DEG	0	Y - ft/s(m/s)	0
Energy - ft2/s2(m2/s2)	0	Z - ft/s(m/s)	0

| OK | CANCEL | HELP |

Figure 18: The Position And Velocity Input Shell

Entry of position and velocity is coupled and the type of entry is determined by the selection of one of five toggle-buttons. If *Earth Centered Inertial Position and Velocity* is chosen, the *ECI Position Vector* text fields are activated and any values in the *All Velocity Vectors* text fields are read as Earth Centered Inertial (ECI) values. *Spherical Position/* provides three options for velocity: *Local Inertial*, *Atmosphere Relative* and *Earth Relative Velocity*. Toggle-button selection determines the reference frame to be applied to the *All Velocity Vectors* text fields. The three components of the position vector are entered using various reference frames depending on toggle selection in the *Spherical Position* section. When *Orbital Parameters* is chosen for initial position and velocity input, all text fields except those in the Orbital Parameters section are inactive.

7.   Type of Propulsion/Throttling

The *Propulsion/Throttling* shell (Figure 19) is a Bulletin Board with four Radio Boxes, fourteen toggle-buttons, four separators, four labels and the three common push-buttons. Selection of any *Propulsion Type* other than the default of *No Thrust* enables the other Radio Boxes for selection. The *Rocket Engine* and *Jet Or Ramjet Engine* toggle-buttons are used to model various single propellant or multi-propellant engines. Selection of any *Throttling Parameter* other than *Do Not Calculate* requires the entry of either polynomial variables or a table name using the file selector. Choosing *Calculate Weights And Volumes* in the *Propellant Weights/Volumes Calculations* section opens the *Vehicle/Propellant Weights* shell (for a description of shell see Section 8 following). The flowrate can be integrated by selecting *Active* in the *Integrate Flowrate of Engine* section.

Up to 15 engines can be defined in the POST program. The complexity of the engine propulsion and weight modeling in POST is such that the present version of G-POST only provides input for the first engine. Engines two through fourteen must be entered in the text entry window manually. The *HELP* push-button provides detailed information to the new user on this data entry method.

Propulsion Type

◆ No Thrust

◇ Rocket Engine

◇ Jet Or Ramjet Engine

◇ Delta-V Addition Using Desired Delta-V

◇ Delta-V Addition Using Current Height

Throttling Parameter

◆ Do Not Calculate

◇ Cubic Polynomial Constant = ETA

◇ Cubic Polynomial Constant = ETAPC(1)

◇ Table Lookup

◇ Linear Function

Propellant Weights/Volumes Calculations

◆ Do Not Calculate

◇ Calculate Weights And Volumes

Integrate Flowrate Of Engine

◆ Inactive

◇ Active

| OK | CANCEL | HELP |

Figure 19: The Propulsion/Throttling Input Shell

8. Vehicle/Propellant Weight Input

A Bulletin Board with Radio Box and Row Column containers, toggle-buttons, labels, separators and the standard three push-buttons, make up the *Vehicle/Propellant Weights* shell (Figure 20). This shell is accessed by using either the *Vehicle/Propellant Weight Input*
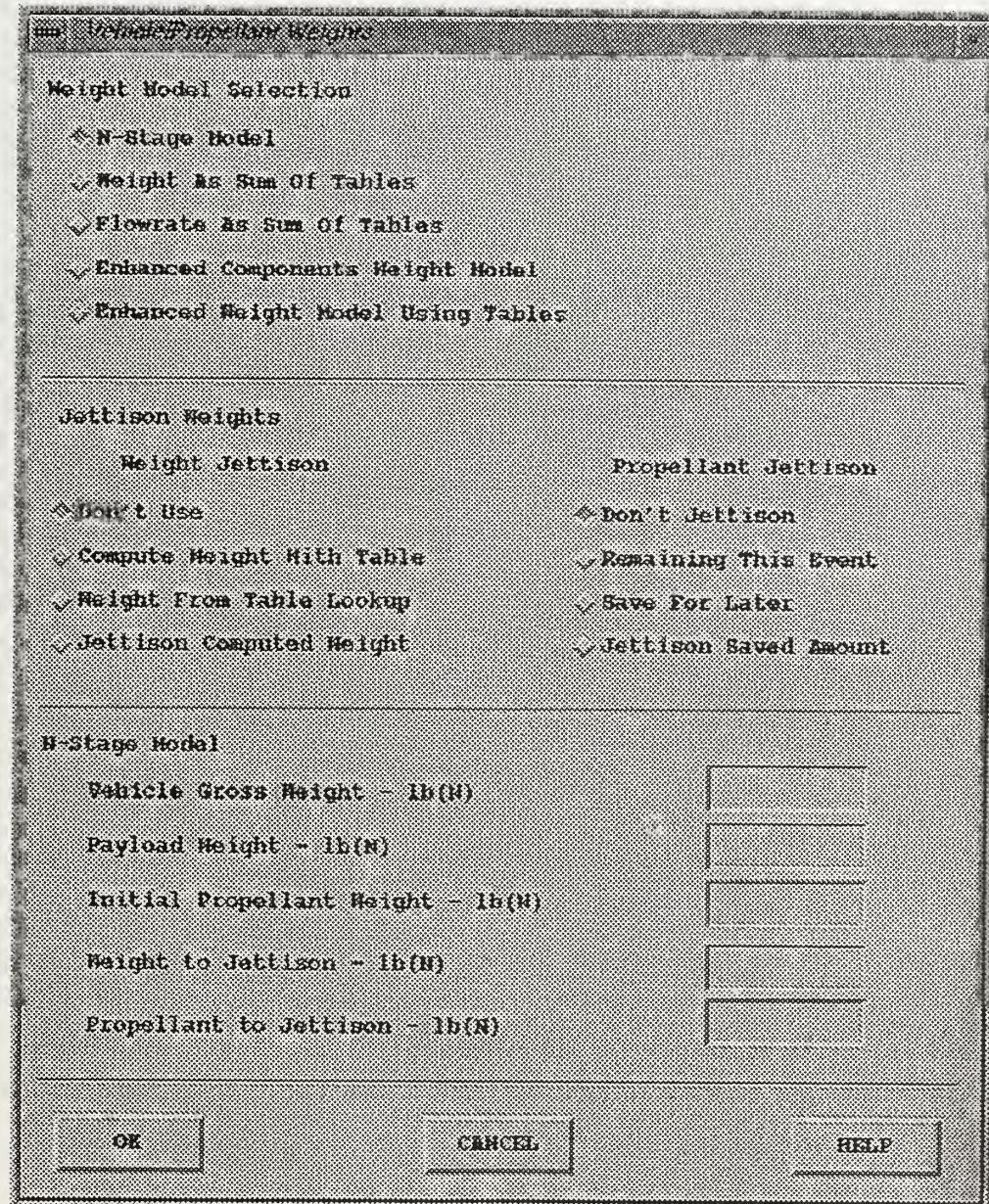
Figure 20: The Vehicle/Propellant Weights Input Shell

push-button on the *Initial Conditions* shell or by selecting *Calculate Weights And Volumes* in the *Propulsion/Throttling* shell. Five selections are available for *Weight Model Selection*; *N-Stage Model*, *Weight As Sum Of Tables*, *Flowrate As Sum Of Tables*, *Enhanced Components Weight Model* and *Enhanced Weight Model Using Tables*. In the *N-Stage Model* all step dry weights and engine propellant weights are lumped together in a single stage weight exclusive of the payload weight. The gross weight of the vehicle at the beginning of each phase can be specified or calculated using the previous phase weight and subtracting any jettison weight and expended propellant. The five text fields are provided for entry using this option.

File Selectors are used to retrieve the table data for the table entry methods. If either the *Enhanced Components Weight Model* or *Enhance Model Using Tables* is selected, the dry weight, jettison weights, total propellant and usable propellant for each step are input individually.

Jettison Weights are selected using the *Weight Jettison* or *Propellant Jettison* toggle-buttons. The method of computing the jettison weight is selected in the initial conditions stage, and the actual jettison is done during a later event.

9.   Method of Guidance

The *Method Of Guidance* shell (Figure 21) is constructed using a Bulletin Board shell holding two Radio Boxes, fifteen toggle-buttons, labels, separators and the three standard push-buttons. *Open/Closed Loop Guidance* is not implemented in this version of G-POST and the corresponding buttons are grayed. Selection of any of the toggle-buttons under *Guidance Type* calls another shell for data entry. The *Aerodynamic Body Rates* shell (Figure 22) is called when the *Vehicle Body Rates* guidance toggle-button is selected. This is a large Bulletin Board Shell containing Radio Box and Row Column containers, toggle-buttons, twenty-two text fields, labels, separators and the standard push-buttons.

Guidance Type

- ⌄ Vehicle Body Rates
- ⌄ Aerodynamic Angles
- ⌄ Inertial Euler Angles
- ⌄ Relative Euler Angles
- ⌄ Inertial Aerodynamic Angles
- ⌄ Inertial Pitch-Plane Angles (PPA)
- ⌄ PPA With Linear Sine Steering

Open/Closed Loop Guidance

- ⌃ NONE
- ⌄ General Open-Loop
- ⌄ General Open-Loop Override
- ⌄ General Closed-Loop
- ⌄ Open-Loop Approx. To Linear Sine
- ⌄ Level Flight Steering
- ⌄ Generalized Acceleration Steering
- ⌄ Predictor-Corrector Guidance

| OK | CANCEL | HELP |

Figure 21: The Guidance Method Shell

37

Method of Calculation

- Inertial Yaw/Pitch/Roll
- Inertial Yaw/Pitch & Bank Angle Rate
- Inertial Yaw/Roll & Angle Of Attack (AOA) Rate
- Inertial Pitch/Roll & Sideslip Rate
- Inertial Yaw & Bank Angle/AOA Rates
- Inertial Roll & AOA/Sideslip Rates
- Inertial Pitch & Bank Angle/Sideslip Rates
- Bank Angle/AOA/Sideslip Rates
- Relative Yaw/Pitch/Roll
- Inertial Yaw/Roll & Drive AOA Rate to Desired
- Constant Body Rate From Criteria
- Constant Body Rate From Quaternian Rate

Body Rate Initialization

- Use Bank Angle/AOA/Sideslip Rates
- Use Inertial Yaw/Pitch/Roll

Initial Values

Yaw or Sideslip	0.0	Azimuth	0
Pitch or AOA	0.0	Latitude	0
Roll or Bank Angle	0.0	Longitude	0
		Body Rate	0.0

Term	Constant	Linear	Quadratic	Cubic	Name
Yaw	0.0	0.0	0.0	0.0	TIMES
Pitch	0.0	0.0	0.0	0.0	TIMES
Roll	0.0	0.0	0.0	0.0	TIMES

OK	CANCEL	HELP

Figure 22: The Body Rates Shell

The toggle-buttons in the *Method of Calculation* section determines the type of variables used in the *Term* text field area and how the body rates are calculated. The *Name* field in the *Term* section is the *Hollerith* name of the variable used in the quadratic polynomial. The default value is TIMES and can be changed by double-clicking in the text field area to display a selection box with the possible variable names (see Figure 24 for an example of a selection box).

The *Initial Values* are entered as *Yaw/Pitch/Roll* or *Sideslip/AOA/Bank Angle*, depending on the *Body Rate Initialization* toggle-button selection. *Azimuth, Latitude* and *Longitude* are entered in the text fields provided. When *Constant Body Rate From Quaternion Rate* is selected in the *Method of Calculation* area, the Body Rate text field is made available for text entry.

If any value other than *Vehicle Body Rates* is selected in the *Guidance Type* section of the *Method of Guidance* shell (Figure 21), the *Angle Guidance* shell is displayed (Figure 23). This shell is another large Bulletin Board containing Radio Box and Row Column containers, thirty-one text fields, eight toggle-buttons, separators, labels and the three standard push-buttons.

Default conditions depend on which of the toggle-buttons was selected in the *Method of Guidance* shell. *Calculate Angles Based On Same Method For All Angles* is the default value in the *Attitude Channel Selector* section for all guidance methods. The two buttons in this section control the text fields in the *Angle Coefficients* section. Only *Yaw/Sideslip* text fields are available for entry in the default condition and these values apply to all channels (*Yaw, Pitch* and *Roll*). When *Calculate Angles With Separate Methods For Each Angle* is selected the other text fields are also available for entry. Values are entered into the polynomial text fields as desired. Double-clicking in the *Method* text field displays the *Method_Popup* selection box dialog (Figure 24). The availability of the *Angle Coefficients, Initial Values* and *Arg Names* text fields is dependant on the *Method* selection. The *Desired Value* and *Event #* default conditions are grayed for all selections of *Method* except 3.

Figure 23: The Angle Guidance Shell

Azimuth, Latitude and *Longitude* are only modifiable if any inertial *Guidance Type* is chosen from the *Method Of Guidance* window. Pitch-plane angle steering must be chosen

before the *Pitch-Plane Angle* and *Kick Over Angle* sections are available. *Pitch Reference* becomes available when *BX Above/Below Local Horizontal* is picked.
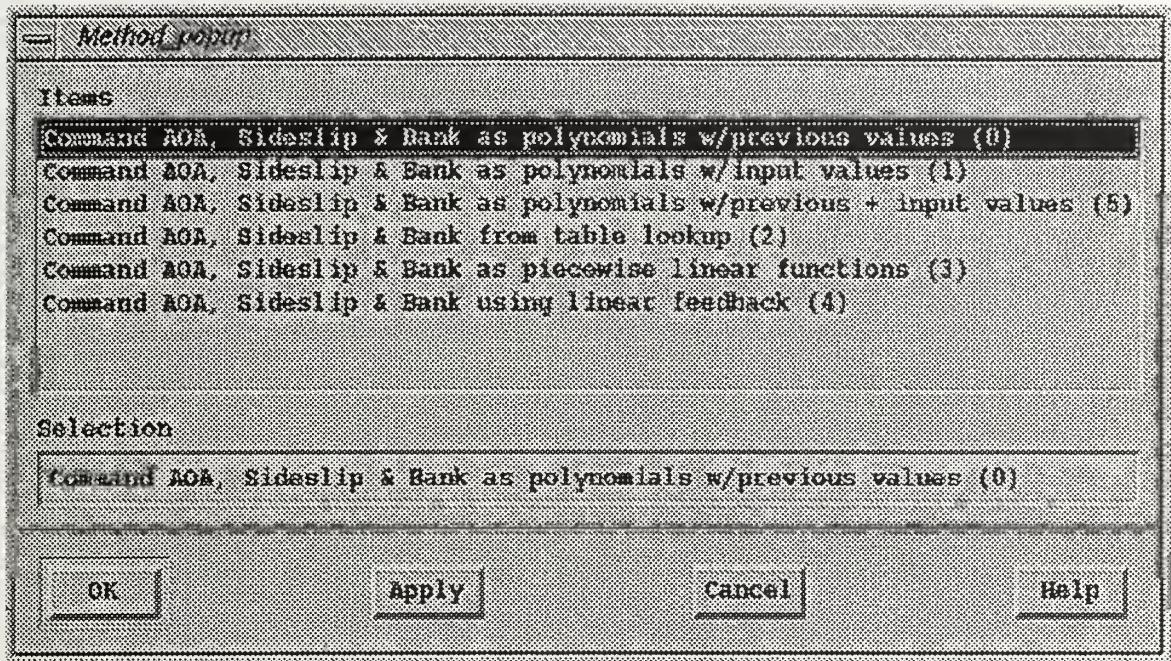
Figure 24: The Method Popup Dialog

10. Print Variable Request

The final selection in the *Initial Conditions* shell is the *Print Variable Request* shell (Figure 25). This shell controls printing of *Input Conditions* and number of output variables, *Trajectory Information, Special Print Blocks, Print Interval*(s), *Number of Lines Per Page, Number of Variables Per Line*, any *Title* to use and *Profil* print options. This is accomplished using a Bulletin Board shell to hold the Radio Box and Row Column containers, text fields, toggle-buttons, labels, separators and the three standard push-buttons.
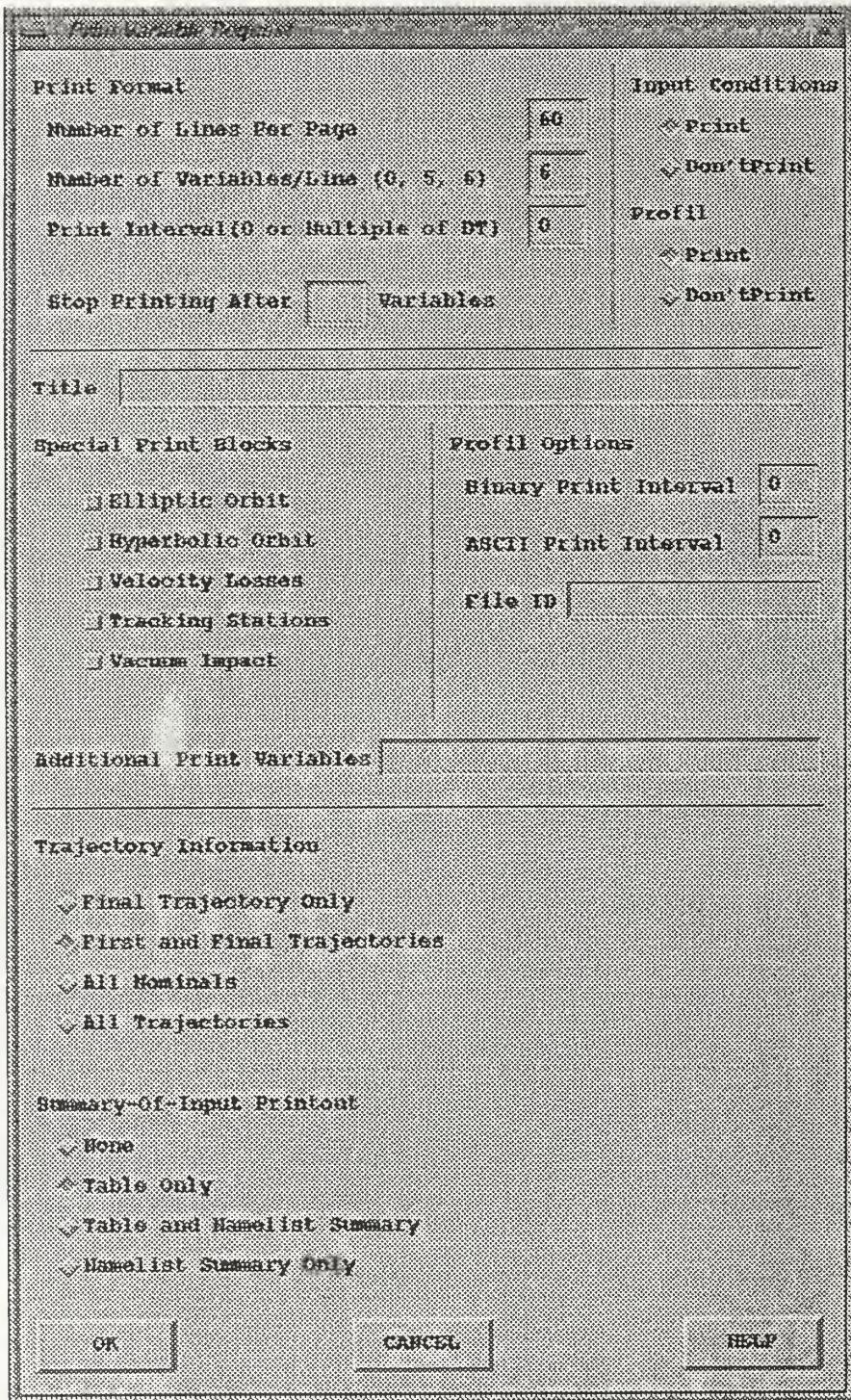
Figure 25: The Print Variable Request Shell

The *Print Variable Request* shell is divided into five areas, consisting of *General* and *Profil* print requests. Selections are made for the desired output using the toggle-buttons and text fields. Larger text field areas are provided for entering a *Title* or *File ID*. Additional print variables are chosen for printing using the *Special Print Blocks* or *Additional Print Variables* printing selection. Double-clicking in the *Additional Print Variables* text field displays a selector popup (like that in Figure 24) for choosing the additional variables to print. Printing of individual events is done in the Event section.

11. Targeting and Optimization

Targeting and optimization specifications are available once the *OK* button is pressed on the *Initial Conditions* shell. The *Target Optimize* button opens the initial shell which consists of a Bulletin Board holding Radio Box and Row Column containers, toggle-buttons, text fields, labels, separators and the three standard push-buttons (Figure 26). Selection of *Projected Gradient* or *Accelerated Projected Gradient* enables entry of values for the *Control*, *Target* and *Optimization Variable* text fields.
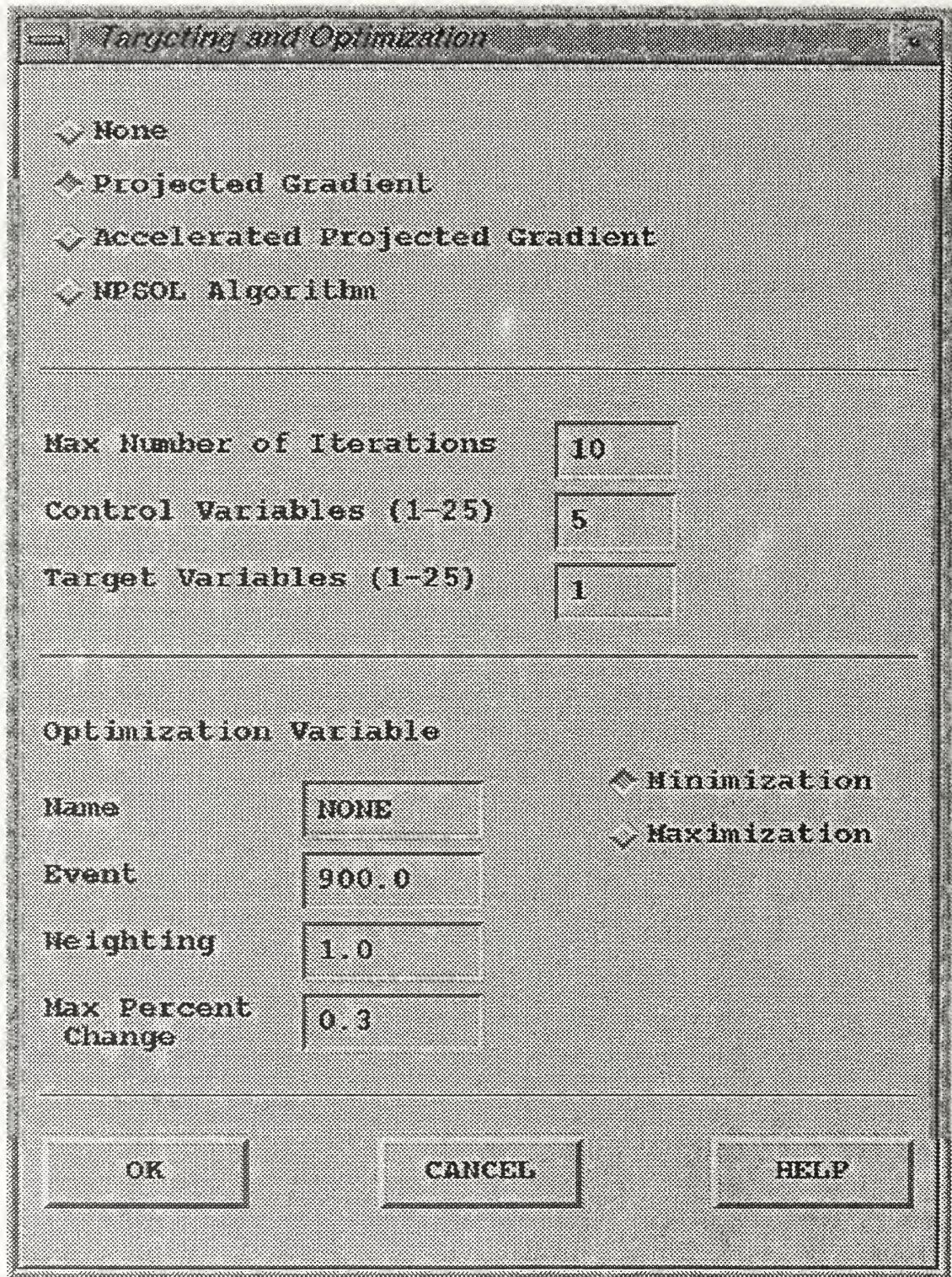
Figure 26: The Targeting And Optimization Shell

44

There can be anywhere from one to twenty-five *Control* or *Target Variables*. The number of variables entered in the text field determines the size of the entry shell that is displayed. The *Control* or *Targeting Parameters* shell (Figure 27) consist of a Bulletin Board holding one to four Row Column containers. These containers are set to display control text fields with four rows and targeting text fields with five rows. The number of

**Figure 27: The Control Parameters Shell
With 5 Entries**

columns changes depending on the number of variables selected. Between one and seven columns are displayed for each Row Column container. Label gadgets are created as necessary to provide organization to the text field entries. A single *OK* push-button is provided and remains at the bottom of the shell no matter what size the shell requires. The *Name* text field in both the *Control* and *Targeting Parameters* shells displays the *Hollerith* variable required by POST. Selection of the variable is made easier by using a selector popup much like the Method Popup shell (Figure 24). Double-clicking in the *Name* text field brings up a the popup shell and the variable can then be chosen using its English description. Direct typing is allowed and values from any text field can be cut, copied or pasted using the *Edit* menu.

The *Optimization Variable* is updated in the same manner as the *Control* and *Targeting Parameters*. Double-clicking on the *Name* text field brings up the same requester as described above. The *Maximize* and *Minimize* toggle-buttons refer to the *Optimization Variable* and are used to designate the type of optimization. Selecting *OK* when completed puts all the variable selections in the multi-line text field of the *Event Worksheet* and enables the Event push-buttons for entry.

12.   Event Push-Buttons

Selection of *Insert* or *Next* displays the *Event Selector* shell (Figure 28) which consists of a Bulletin Board containing a Radio Box with two toggle-buttons, four text fields, labels and the three standard push-buttons. All Event inputs, after the *Initial Conditions*, are made
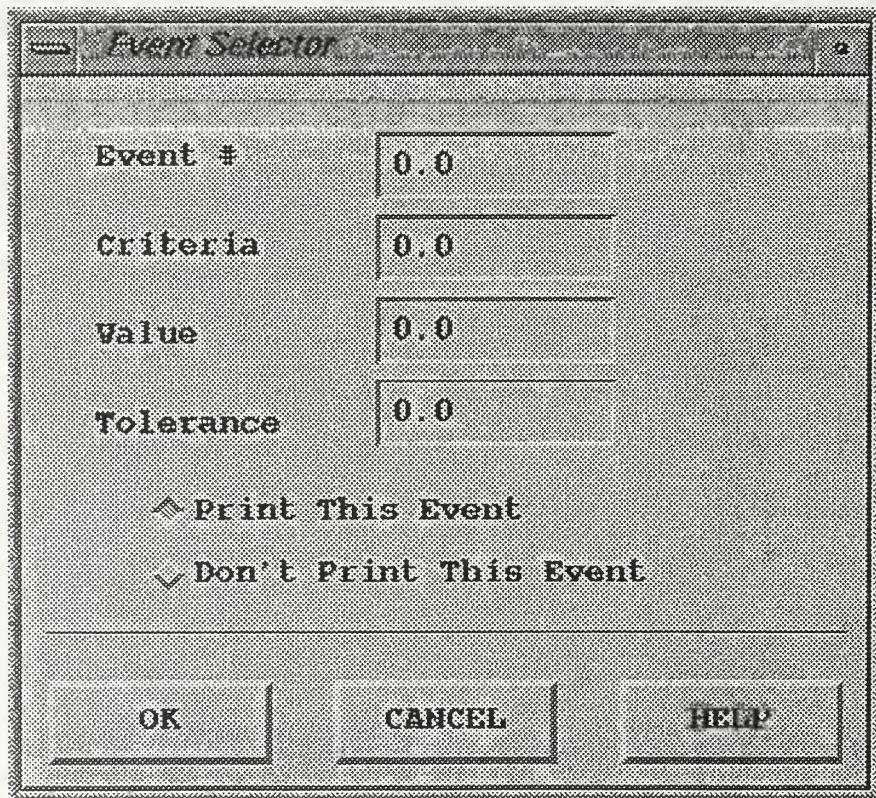
Figure 28: The Event Selector Shell

using this shell and the eight Event selection push-buttons as described above in Section **D**. The next available Event number is displayed in the *Event #* text field if the *Next* button

was pushed, or blank if *Insert* was pushed. Putting a number in the *Event #* text field will insert the Event selections at the appropriate position in the input file. All Event numbers are adjusted to reflect any changes due to an *Insert* operation. The current Event *Criteria* and *Value* are entered in their respective text fields and a print selection can be made using the *Print This Event* toggle-buttons.

While the *Event Selector* shell is open, all selections from the eight Event selection push-buttons are entered under that Event number. Selecting any button other than *Other Controls* displays the appropriate shell as described in the previous sections. *Other Controls* is a button for added features and is not implemented at this time. When the *OK* button on the *Event Selector* shell is pressed, all selections made while the shell was open are written to the multi-line text field. If mistakes were made, *Delete* can be used to remove the entire event or the editing features can be used to correct the error.

When the final Event entry has been made, the *DONE* button is used to display the *Last Entry Dialog_popup* (Figure 29). Selecting *NO* from this dialog box saves the file and closes the *Event Worksheet* shell. Selecting *YES* displays the *Trajectory Abort Specifications* shell (Figure 30), which consists of a Bulletin Board shell holding three text field widgets, their labels and the three standard push-buttons. These values can be edited or left at their default values. Selecting OK adds these values to the text entry area of the *Event Worksheet*, saves the program and closes the *Event Worksheet* shell.

Figure 29: The Last Entry Dialog

47
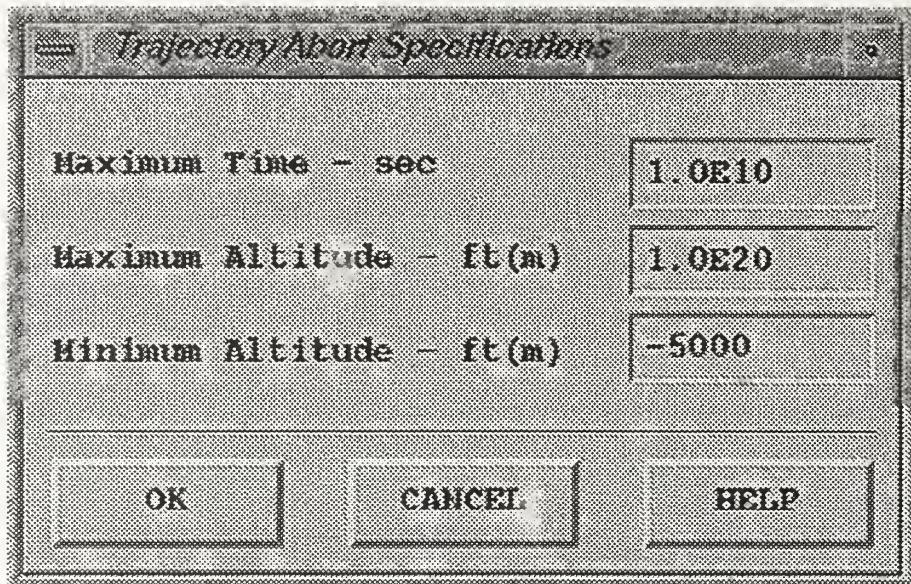
Figure 30: The Trajectory Abort Shell

Another file can then be edited if desired, or any of the other options on the *G-POST* shell selected. A POST input file can be run through the POST program with or without the *Event Worksheet* being open.

E. RUNNING A POST FILE

One or more output files, depending on print selections, are created using this option. This option will only work if POST can be run from the same location G-POST is started form. When *Run Post File* is selected, the open file requester appears (Figure 6) with the filter set to *.inp. After choosing the desired file and clicking on *OK*, another requester appears asking for the name of the output file. The default value is the input file with a .out extension. Selecting OK will run the selected input file through POST to obtain the output files. Any errors encountered will appear in the UNIX shell and can be corrected using the *Event Worksheet* editing tools. Once the output file is obtained, either of the next two push-buttons on the *G-POST* shell can be executed.

F. ANALYZING DATA

As mentioned in Chapter II, the data from the POST output file must be converted to a useful format prior to any analysis. When the *Analyze Data* push-button is selected, a file selector is presented requesting the *Profila* file name. The desired file can be selected by clicking on the *OK* button. As in the previous section, a requester is displayed with the default file name the same as the *Profila* file except for the .mat ending. This Matlab file is produced by clicking on the *OK* button. The Matlab program will then start in the UNIX shell on those machines that have the program installed. The file will be loaded and ready for analysis. The *HELP* button on the *G-POST* shell yields more information on how to obtain data from this new file.

G. RUNNING A 3D SIMULATION

Selecting Run Simulation displays a file requester that needs to find a *Profila* to run correctly. If an incorrect file is selected the program will load the default file, Example Problem 1 [Ref 1], and display the *Visual POST* window (Figure 31). A new file can be opened at any time using the appropriate pull down menu. Other options are provided as explained by the Help function in the *Visual POST* program. This program is provided as a prototype of future trajectory simulation projects.
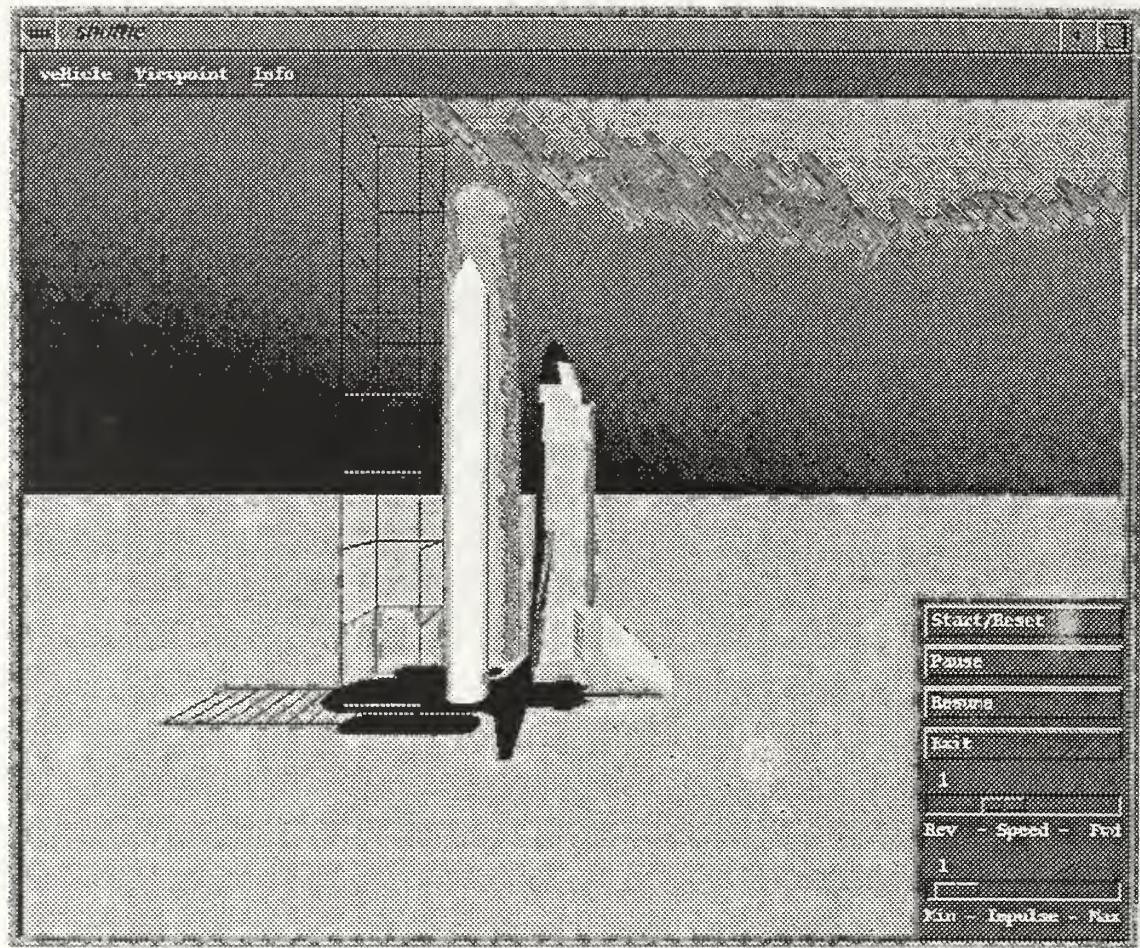
Figure 31: The Visual POST Program

IV. INTERFACE IMPLEMENTATION - A TUTORIAL

To demonstrate the ease of using G-POST, the Example Problem 1 in the POST Utilization Manual, pg. 8.a.1 [Ref 1], is used to build a POST input file. The example is described here followed by the corresponding G-POST input requirements. The entire problem definition is reproduced in Appendix B.

A. EXAMPLE PROBLEM 1

An important ascent trajectory optimization problem during the conceptual phases of the Space Shuttle program was that of determining the maximum payload capability of various configuration concepts. One such Space Shuttle configuration is represented by this sample problem. The four key components of this configuration are the orbiter, two solid rocket boosters (SRBs), and the external tank (ET). This multibody nonsymmetrical configuration created special simulation requirements that motivated many of the features contained in POST. For example, to accurately predict the performance capability of a unsymmetrical configuration such as Space Shuttle, it is important to include the thrust vectoring losses encountered as the engines gimbal to balance the aerodynamic moments caused by the nonsymmetrical shape of the configuration. This fact led to the development of the static trim option employed in this sample case.

B. EVENT WORKSHEET ENTRY

For this problem we will be generating a new file (tutorial.inp) by inputting information into the Event Worksheet in three steps: Initial Conditions Input, Targeting and Optimization Input and Event Input. An example of the text output will be presented after each step. The input data listing for this sample case is presented in Appendix C for comparison with the data file generated using G-POST (Appendix D). All tables used in

the example were put in individual files with .tab extensions (e.g., vacuum thrust table for engine 1 - vacuum.tab). A complete listing of tables and their file names are found in Appendix E.

1. Initial Conditions Input

After starting the program by typing *gpost* at the UNIX prompt, select *Event Worksheet* from the G-POST window. Choose *Create New File* from the file selection pop-up when the window appears and double-click on *NoName* with left mouse button to highlight it. Replace *NoName* by typing *tutorial.inp* and then select the *OK* button. The *Event Worksheet* appears with the file name along the title bar (like Figure 9, pg. 20). *Initial Conditions, Done* and *Help* are the only buttons that can be selected at this time. The text area is available for typing into or copying from another open file.

Select

to get started.

The first button in the *Initial Conditions* window (Figure 10, pg. 21) is for specifying whether English or metric units are to be used for input and output (Figure 11, pg. 22). The example problem uses English units for both, so the default value of *English/English* is chosen (Note: It is not necessary to select any of the default values, they will be chosen automatically if no modifications are made).

Aerodynamic Coefficients and *Static Trim* options are selected using the *Aerodynamic Inputs* window (Figure 12, pg. 24). The example problem requires input of drag, lift and static trim coefficients. The forces are specified by selecting the third toggle button, *Drag and Lift Force Coefficients*, and entering the desired table name in the file requester. All files with the .tab ending in the current directory are shown in the *File* area. Select **dragforce.tab** and click on *OK* to accept choice. The file selector reappears asking for the name of the next Forces table. Select **liftforce.tab** and click on *OK* to accept choice. The order of table entry is not important. There are only two tables in this example. Select

cancel to exit the file requester. If it were necessary to enter more tables, continue selecting files in the same manner until they are all entered.

Next, select *Static Trim Aerodynamic Coefficients* button to bring up the window for the static trim selections (Figure 13, pg. 25). Select *Pitch Only* for static trim type and enter the engine gimbal locations in body reference coordinates for the engine. The coordinates are in feet and entered as follows: $X = \mathbf{218.42}$, $Y = \mathbf{0.0}$, and $Z = \mathbf{33.33}$. The last thing to do in the *Static Trim* window is to enter the force *Reference Dimensions* for *Length* and *Area* in feet and square feet. These variables are included in the calculations when using the static trim option. The *Area* is equal to **4500.00** and the *Length* is equal to **218.833** (no entry is made in *Length In Yaw*). Click on *OK* to accept the entries and exit static trim selection. At this time the program will ask for table entry. As with the force coefficient selections described earlier, select the files for as many tables as need to be entered. The help function for the window calling the file selector describes which tables are necessary. For this example select **momentcoeff.tab** and **aeroref.tab**. Click on *CANCEL* after these two selections have been made to tell the program there are no more tables to be entered. Select *OK* when all *Aerodynamic Inputs* are complete.

The selection of the desired *Numerical Integration Methods* (Figure 14, pg. 27) follows the *Aerodynamic Inputs*. From the *Methods of Integration* window choose **4th Order Runge-Kutta** from the *Integration Methods* and increase *Step Size* to **5.0**. Select *OK* to accept the choices and close the window.

Atmospheric/Gravity Models selection is next. Default settings are required for *Atmosphere Model, Winds, Aeroheating*, and *Special Aeroheating* (like Figure 15, pg. 29). Change the *Gravity Model* from *Oblate Planet* to *Spherical Planet*. Select *OK* or *CANCEL* to continue specification of *Initial Position/Velocity* values. The *Position and Velocity Input* shell (Figure 18, pg. 32) is opened with the default value of *Initial Position And Velocity* set to *Spherical Position/Earth Relative Velocity*. No other entries are required. The values for position will be entered in the *Angle Guidance* window. The initial velocity of the Space Shuttle on the launch pad is zero. Select *OK* to close the window.

Selecting *Type of Propulsion/Throttling* displays the *Propulsion/Throttling* window (Figure 19, pg. 34). Select *Rocket Engine* from *Propulsion Type* and choose the required tables using the previously described table entry method. The required tables for this example are **vacuum.tab** and **exitarea.tab**. Select *Calculate Weights And Volumes* from the *Propellant Weights/Volumes Calculations* area. This selection calls the *Vehicle/ Propellant Weights* window (Figure 20, pg. 35). Input the *Initial Propellant Weight* in the *N-Stage Model* section as **2249000.0**. Click on *OK* to accept changes and close this window. Next click on *OK* to close the *Propulsion/Throttling* window.

Since the current version of G-POST only provides limited input tools, and for just one engine, the flowrate (IWDF(i)) and impulse (ISPV) must be typed into the text entry area of the *Event Worksheet*. This will be done later in the tutorial, after *OK* has been pressed on the *Initial Conditions* window. The number of engines will automatically be set to "1" (NENG = 1,) and printed out. This number can be changed using the text editing tools, and other information can be added for this or additional engines by following the *HELP* information on the *Propulsion/Throttling* window. But, for the purpose of this example only one engine is required.

The *Method of Guidance* window (Figure 21, pg. 37) requires the selection of a *Guidance Type* from the choice of seven toggle buttons. Select the third button, *Inertial Euler Angles*, to display the *Angle Guidance* window (Figure 23, pg. 40). The X, Y and Z components of guidance (channels) can be controlled using the same method for all channels, or separate methods for each channel. The default, and the value used in this example, is to *Calculate Angles Based On Same Method For All Angles*. In the *Angle Coefficients* section double click in the *Method* text field next to *Yaw/Sideslip* to bring up the popup for the *Method* type (Figure 24, pg. 41). Click on the second entry to select **Command AOA, Sideslip & Bank as polynomials w/input values(1)**. The value in the Method text field should be **1**. Polynomial coefficients for the three components would be entered at this time if required. *Inertial Azimuth, Latitude* and *Longitude* are entered as

90.0, **28.5** and **279.4** respectively in the *Angle Guidance* window. Click on *OK* to close the *Angle Guidance* window and again to close the *Guidance Method* window.

The last selection on the *Initial Conditions* window displays the *Print Variable Request* window (Figure 25, pg. 42). The *Print Interval* is set to **20.0** in the *Print Format* section and the title, **sample problem for ascent trajectory w/ drop tank orbiter,** is entered into the *Title* text field. All other entries are left at their default values or blank. Click on OK to store these values and close the window.

Once all the initial conditions have been entered, Click *OK* on the *Initial Conditions* window to print all of these values to the *Event Worksheet* multi-line text entry area and close the *Initial Conditions* window. Figure 32, pg. 56, shows the state of the multi-line text entry area after OK was pressed. At this time enter **IWDF(1) = 2,** and **ISPV = 439.0,** following NENG = 1, in the multi-line text entry area. The Target Optimize button is now available for selection.

Initial Conditions | Integration Method | Propulsion Type | Atmosphere Model | Print Options

Target Optimize | Guidance Setting | Aerodynamic Coefficient | Target Input | Other Constants

INSERT | DELETE | NEXT | PREVIOUS | DONE | HELP

```
1$GENDAT
  PRNC     =   0,                    / Profil binary print interval
  PRNCA    =   0,                    / Profil ASCII print interval
  PINC     =   20,                   / Print interval
  TITLE    =   OH*sample problem for ascent trajectory w/ drop tank orbite
  EVENT    =   1,
  NPC(2)   =   1,                    / Runge-Kutta integration
  DT       =   5.0,                  / Step size
  EPSINT   =   0.01,                 / Error tolerance
  NPC(5)   =   2,                    / 1962 US atmosphere
  NPC(26)  =   0,                    / Aeroheating flag.
  NPC(16)  =   1,                    / Oblate planet gravity model
  NPC(3)   =   4,                    / Velocity vector
  NPC(4)   =   2,                    / Spherical position vector
  AZL      =   0,                    / Azimuth
  GDLAT    =   0,                    / Geodetic latitude
  LONG     =   0,                    / Relative longitude
  NPC(8)   =   2,                    / Drag and lift coefficients
  GXP      =   218.42,               / X location of engine gimbal
  GYP      =   0.0,                  / Y location of engine gimbal
  GZP      =   33.33,                / Z location of engine gimbal
  SREF     =   4500.00,              / Reference area
  LREF     =   218.833,              / Reference length
  NPC(9)   =   1,                    / Rocket propulsion
  NENG     =   1,                    / Number of engines
  IWDF(1)  =   2,                    / Engine flowrate
  ISPV     =   439.0,                / Engine impulse
  WPROPI   =   2249000.0,            / Initial propellant weight
  NPC(21)  =   1,                    / Calculate propellant weights
  IGUID(1) =   1,                    / Inertial Euler angles guidance
  IGUID(4) =   1,                    / Polynomials w/input values
  MAXTIM   =   1000.0,               / Maximum time before termination
  ALTMAX   =   2000000.0,            / Maximum altitude before terminati
  FESN     =   12,                   / Final event number
  $
1$TBLMLT
  $
1$TAB
  TABLE    =   6htvclt ,0,5472000.0,
  $
1$TAB
  TABLE    =   6haelt  ,0,232.5,
  $
1$TAB
  TABLE    =   6hcdt   ,2.6hmach  ,6halpha ,12.5.1.1.1.1.1.1.1.1,
  -20.,
```

Figure 32: Event Worksheet After Initial Conditions

56

2. Targeting and Optimization Input

Select

Target
Optimize

to display the *Targeting and Optimization* window (Figure 26, pg. 44). Select *Projected Gradient* and leave the default value for *Max Number of Iterations* at **10**. Enter **9** in the *Control Variables* text field area and press the enter key. In the *Control Parameters* window (Figure 27, pg. 45) double-click in the first *Name* text field. Select **Vehicle Gross Weight (6HWGTSG)** from the popup selector and press *OK*. Enter **1** for *Event*, **4031000.0** for *Initial Guess* and **1.0** for *Pert*. Use the same method to choose **Pitch Angle Linear (Rate) Term (6HPITPC2)** as the *Name* field for the remaining eight entries. Enter the remaining values as described in Table 1. Select *OK* to save the information.

Entry #	Event	Initial Guess	Pert
2	2	-1.8	1.0
3	3	-.5	1.0
4	4	-.2	1.0
5	5	-.3	1.0
6	6	-.25	1.0
7	7	-.3	1.0
8	9	-.15	1.0
9	10	-.05	1.0

Table 1: Entry Values for Control Parameters Window

Repeat the steps for the *Target Variables* (Figure 26, pg. 44) by first entering a value of **3** in the *Target Variables* text field. Enter the values for *Name, Value* and *Tolerance* as

described in Table 2. The other fields in this window can be left at default values. Click on *OK* to save values and close window.

Entry #	Name	Value	Tolerance
1	Altitude Above Oblate Planet (6HALTITO)	303805.0	100.0
2	Inertial Velocity (6HVELI)	25853.0	.1
3	Inertial Flight Path Angle (6HGAMMAI)	0.0	.001

Table 2: Entry Values for Targeting Parameters Window

The *Optimization Variable* values are entered in much the same manner as the *Control* and *Target Variables*. Double-click on the *Name* text field to display the selector popup. Choose **Current Weight of Vehicle (6HWEIGHT)** and click on *OK*. Enter **12.0** for the *Event* and **1.0E-6** for the *Weighting*. The *Max Percent Change* is left at its default value. Select the *Maximization* toggle-button and then the *OK* button to close the window and print the selections into the multi-line text entry area of the *Event Worksheet*. Figure 33, pg. 59, shows the *Event Worksheet* after selecting *OK* on the *Targeting and Optimization* Window. The Event selection buttons are now available for selection.

Figure 33: Event Worksheet After Targeting And Optimization Input

3. Event Input

The remaining events are entered sequentially from Event 2 to Event 12. Selection of

NEXT or INSERT

displays the *Event Selector* window (Figure 28, pg. 46). Click on *Next* to bring up the window. The number in the *Event #* text field will read **2**. Double-click on the Criteria text field to bring up a selector popup. Choose **Time (6HTIME)** and then click on *OK*. Next enter **15.0** for the *Value* field. While the *Event Selector* window is still open, select the *Guidance Method* button from the *Event Worksheet*. Once again select *Inertial Euler Angles* from the *Guidance Type* section. Double-click on the Method text field and change the value to **Command AOA, Sideslip & Bank as polynomials w/previous values (0).** Close the two guidance windows by clicking on *OK*. Click on *OK* in the *Event Selector* to print the values to the *Event Worksheet* multi-line text entry area. The above steps are repeated for events 3 through 12. Table 3 describes the required input values for the *Event Selector* text fields of events 3 through 12. Any additional input selections are made while the respective Event Selector window is open. These selections are as follows:

• **Event 7** - Select the *Integration Method* button and change the *Step Size* from 5.0 to **10.0.**

• **Event 8** - Select the *Propulsion Type* button and change *Propulsion Type* from *Rocket Engine* to *No Thrust*. Select the *Aerodynamic Coefficient* button and de-select *Static Trim Aerodynamic Coefficients*. After selecting *OK* from the *Event Selector*, an addition must be made to the text entry area of the *Event Worksheet*. Add **WEICON = 0.0,** before ENDPHS.

• **Event 9** - Select the *Integration Method* button and change the *Step Size* from 20.0 to **50.0.** Select the *Print Options* button and change *Print Interval* from 10.0 to **20.0.** Select the *Propulsion Type* button and change *Propulsion Type* from *No Thrust* to *Rocket Engine*. Enter the **vacuum2.tab** and **exitarea2.tab** tables using the file selector. Select

Aerodynamic Coefficient button and select *Static Trim Aerodynamic Coefficients* to display the *Static Trim* window. Change the *Engine Gimbal Location* to **142.0**, **0.0** and **25.0** for X, Y, and Z respectively. Change the *Reference Area* to **4840.0** and the *Reference Length* to **135.0**. Enter **dragforce2.tab** and **liftforce2.tab** for the *Aerodynamic Coefficient* window and **xcentergrav.tab, ycentergrav.tab** and **zcentergrav.tab** for the *Static Trim* window using the file selector as it appears when *OK* is pressed. After selecting *OK* from the *Event Selector*, an addition must be made to the text entry area of the *Event Worksheet*. Add **WJETT = 665000.0,, WPROPI = 809000.0,** and **ISPV = 459.0,** before GXP.

Event #	Criteria	Value	Tolerance
3	Time (6HTIME)	25.0	0.3
4	Time (6HTIME)	40.0	0.3
5	Time (6HTIME)	60.0	0.3
6	Time (6HTIME)	120.0	0.3
7	Time (6HTIME)	150.0	0.3
8	Weight of Remaining Propellant (6HWPROP)	0.0	2.0E-6
9	Time Since Last Primary Event (6HTDURP)	7.	1.0E-6
10	Time Since Last Primary Event (6HTDURP)	100.0	0.3
11	Time Since Last Primary Event (6HTDURP)	150	0.3
12	Weight of Remaining Propellant (6HWPROP)	0.	0.3

Table 3: Event Selector Text Field Values

• **Event 11** - Select the *Integration Method* button and change the *Conic Calculations* from None to **Calculate At End Of Integration Step**.

After selecting *OK* for Event 12, select *DONE* form the *Event Worksheet*. The requester asks if this is the final event (Figure 29, pg. 47). Choose *YES* and enter the *Maximum Time* as **1000.0**, *Maximum Altitude* as **2000000.0** and leave *Minimum Altitude* at default value for the *Trajectory Abort Specifications* (Figure 30, pg. 48). Select *OK* to save the input file. The input file is now complete and can be run through the POST program.

C. RUNNING A POST FILE

Select *Run POST File* from the *G-POST* window and choose **tutorial.inp** from the file requester. Click on *OK* and enter **tutorial.out** in the second requester when it is displayed. Click on *OK* to obtain the POST output file (*tutorial.out*), the *profila* file (*tutorial.pro*), and the *profil* file. Run time for *tutorial.inp* is approximately 40 seconds on a Sparc 10 workstation. If there had been any errors in *tutorial.inp* these would have been written to the *tutorial.out* file instead of the trajectory information. Run time in this instance is much shorter, approximately 15 seconds. The *profila* information can now be used to generate a Matlab file for trajectory analysis.

(Note: For the above procedure to work, the POST program must reside on or be linked to the SGI workstation running G-POST and have as its execution command *post < inputfilename > outputfilename*. If this is not the case, the POST output file can be obtained by using the UNIX shell of the machine where POST is accessed. Consult the system manager for the location of POST and how to use the UNIX shell, and *A Primer for POST* [Ref 4] for how to operate the POST program.)

D. ANALYZING DATA

The next task is to convert *tutorial.pro* to a format that can be used by Matlab for further analysis. Select *Analyze Data* from the *G-POST* window and choose **tutorial.pro** from the file requester. Click on *OK* to make the file *tutorial.mat* and start the Matlab program (if available on the machine running G-POST). At the UNIX shell command line type **load tutorial.mat** and press enter. Type **who** at the command prompt to display the variables for which there is data. Plotting of the variables can now be done using Matlab

plot commands. Figure 34 shows a plot of altitude versus time for this trajectory. Consult the Matlab manual for other plot commands.
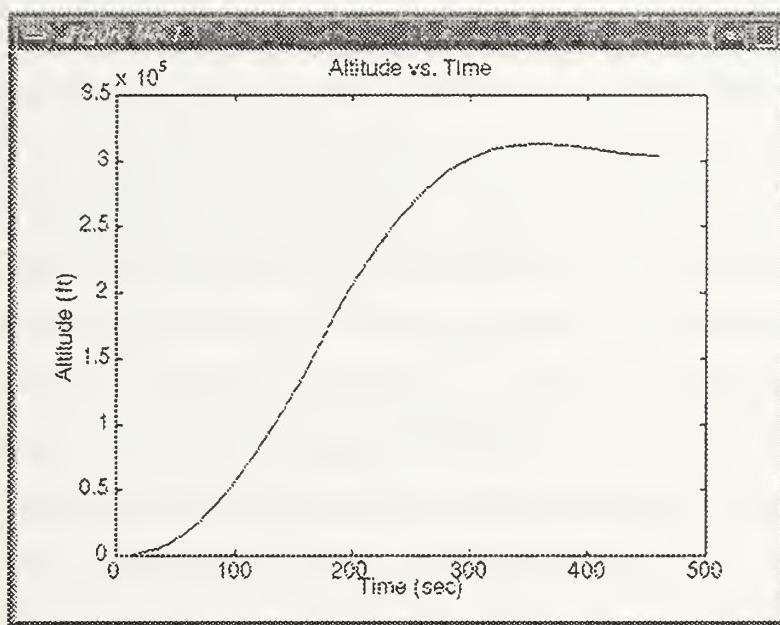
Figure 34: Example of Matlab Plot

E. RUNNING A 3D SIMULATION

Select *Run Simulation* from the *G-POST* window and choose **tutorial.pro** from the file selector. After clicking on *OK*, Figure 31, pg. 50, is displayed with the Shuttle in its launch position. Click on *Viewpoint*, or press the *v* key, and change viewpoint to *Medium*. Select the *Start/Reset* button to launch the vehicle. Use the *Speed* slider to modify the display rate. Select *Near* while the vehicle is moving to follow the Shuttle in its trajectory.

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

POST is a widely-used program to optimize trajectories. Although the Naval Postgraduate School and other aerospace institutions would benefit from the simulation power of POST, the many weeks required to learn the POST input language is too prohibitive for immediate use. The objective of this thesis was to develop a means of making the creation of an input file for POST as straightforward as possible, thus unlocking the power of the program for all users. To meet this objective, a Graphical User Interface (GUI) was developed to guide the user through the proper entry procedure using English terms and default values. Written using the OSF/Motif windowing language, this GUI provides a "point and click" environment that greatly increases the usability of POST. Although compiled on the Silicon Graphics (SGI) workstation, the use of Motif makes the GUI platform independent.

Developing windows for a GUI is considerably easier when using a graphically based design tool. Many hours are saved using programs such as Builder Xcessories to create the initial windows. However, complete reliance on any tool to produce a complex GUI is not advised. A design tool can only accomplish so much in the creation of a complete project. Interactions between all facets of the program must be considered early in the design process. Many false starts can be avoided with an in-depth knowledge of the Motif language. There are numerous methods of obtaining and displaying required user information. Some methods are more difficult to implement than others, and the optimum configuration is not readily obtained without adequate knowledge in window design.

Along with knowing how the windowing design language is best used, a thorough understanding of how the program operates is essential. POST has many options that are accessible only when one or two other options are selected. Mapping out the interdependencies of each variable in the program allows for a more systematic approach to the programming of the GUI. Consultations with current users also provides in-roads

into the thought process used in implementing POST, as well as a better understanding of how a GUI would best suit the user. Frequent review by the end-user ensures that the final product is designed for his use and not that of the programmer.

B. FUTURE WORK

As is usually the case in computer software development, the current version of this product has room for improvement. Although the increase in usability of POST under G-POST is such that long weeks of studying to learn the POST input language is no longer necessary, there are specific areas of work that would benefit users of all levels.

1. Propulsion

POST allows for the use of up to fifteen engines and stages. The propulsion and weight characteristics of the aerospace vehicle can be simulated with as much complexity as desired. Single, multiple or no engines can be applied to an individual stage. The current version of G-POST is limited to one engine and many of the corresponding propulsion variables must be entered manually. Future work would involve determining the optimum input method for this complex procedure, while providing the checks to insure proper entry.

2. Multiple Files

Often times having the ability to open and edit multiple files is advantageous. Opening files side-by-side can facilitate comparison and editing operations. The addition of this feature would require the proper functions to determine which window is primary for input and set that window active. Dynamic generation of either a maximum number of new windows or an unlimited number of windows would also need to be determined.

3. Default Units

The units of input and output can be specified as either English or metric. The default values of all variables are in their English form. Addition of a feature to update defaults to reflect chosen values would be beneficial. Conversion factors are also available in POST and can be edited. Windows could be added to G-POST to allow for these changes.

4. Tables

Construction of the many input tables required by POST can be nearly as complex as the program itself. All tables are assumed to exist prior to G-POST usage. A construction tool to facilitate table construction would increase productivity substantially.

5. Open/close Loop Guidance

POST makes available for use some general open and closed loop guidance laws in addition to the guidance methods provided by G-POST. Although the choices have been added to the guidance window, these options are unavailable in this version.

6. Visual Post

Currently, the 3D visual representation of the POST output file is limited to using inertial coordinates as input, starting with initial position as the launch pad and having only two objects available for visualization; the Space Shuttle and a generic missile. Adding to the capabilities is not a difficult matter and would greatly increase the usefulness of the program.

APPENDIX A. LISTING OF NPC AND IGUID CODES

This appendix contains the NPC and IGUID codes used in a POST input file. They are ordered numerically with related topics noted. Default values are in **bold**. These are provided as a learning tool and guide for checking the G-POST output for desired values.

NPC CODES

NPC(1) - Keplerian conic calculation flag. Section 6.a.5

Value	Definition
0	**Do not calculate conic parameters.**
1	Calculate conic at the end of each integration step but do not print conic print block.
2	Calculate and print conic block only at phase changes.
3	Calculate conic at the end of each integration step and print conic block with each printout.

Related: NPC (31) - for LANVE

NPC(2) - The integration method flag. Section 6.a.15

Value	Definition
1	**Fourth order Runge-Kutta.**
2	Variable step/order predictor corrector.
3	Laplace conic integration. Should be used for integrating orbits about a spherical planet, i.e., if J2-J8 are all zero.
4	Enke-S integration method. Should be used for integrating orbits about an oblate planet.

Related: NPC (20) - Type of special integration step size.

NPC(3) - The velocity vector initialization flag. Section 6.a.12

Value	Definition
1	Input Earth-centered inertial components, VXI(j), j=1,3.
2	Input inertial components in the local horizontal (G) frame, GAMMAI, VELI, and AZVELI
3	Input atmospheric relative components in the local horizontal (G) frame, GAMMAA, VELA, and AZVELA
4	**Input Earth relative components in the local horizontal (G) frame, GAMMAR, VELR, and AZVELR**
5	Input orbital parameters, ALTP, ALTA, INC, LAN, ARGP, and TRUAN. NPC(4) is not used.

Related: NPC (4)

NPC(4) - The position vector initialization flag. Section 6.a.12

Value	Definition
-1	Calculate XI(2) if XI(1), XI(3), and GCRAD are specified. Sign of XI(2) is opposite of VXI(2)
1	Input Earth-centered inertial components, XI(j), j=1,3.
2	**Input spherical coordinates, geocentric or geodetic latitude, GCLAT or GDLAT, relative or inertial longitude, LONG or LONGI, and oblate altitude or geocentric radius, ALTITO or GCRAD.**

Related: NPC (3)

NPC(5) - Atmosphere model flag.   Section 6.a.4

Value	Definition
0	No atmosphere.
1	General atmosphere using inputtables PREST, ATEMT, CST, DENST, and constantsATMOSK(i),i=1,2.Also allows for atmospheric perturbation tables PRESPERT, ATEMPERT, and DENSPERT.
2	**1962 U.S. standard atmosphere.**
3	1963 Patrick AFB atmosphere.
4	1971 Vandenberg AFB atmosphere.
5	1976 U.S. standard atmosphere.
6	General atmosphere using input tables ln(PREST), ATEMT, CST, ln(DENST), and constants ATMOSK(i), i=1,2.Also allows for atmospheric perturbation tables PRESPERT,ATEMPERT,and DENSPERT.
7	Use random atmosphere model.

Related: NPC (6) - Atmospheric winds, NPC (38) - Atmospheric turbulences, NPC (39) - Gusts

NPC(6) - Atmospheric winds flag.   Section 6.a.4

Value	Definition
0	**No winds.**
1	Winds defined by tables of wind speed (VWT), wind azimuth (AZWT), and vertical component (VWWT). VWWT is positive downward.
2	Winds defined by tables of northerly (VWUT), easterly (VWVT), and vertical (VWWT) components. VWWT is positive downward.

NPC(7) - Acceleration limit option flag. Section 6.a.18

Value	Definition
0	**No acceleration limit.**
1	Limit to ASMAX by calculating ETAC.

Related: used if NPC(9) = 1 or 2 and NPC(30) = 0,3,or 4

NPC(8) - The aerodynamic coefficient type flag. Section 6.a.1

Value	Definition
0	No aerodynamic coefficients
1	**Input tables of axial force (CAOT and CAT), normal force (CNOT and CNAT), and side force (CYOT and CYBT) coefficients.**
2	Input tables of drag force (CDOT and CDT), lift force (CLOT and CLT), and side force (CYOT and CYBT) coefficients.
4	Same as option NPC(8)=2, except that viscous aero corrections are added to CL and CD.

Related: NPC (10) - Static trim

NPC(9) - The propulsion type selection flag. Section 6.a.13

Value	Definition
0	**No thrust or velocity addition.**
1	Rocket engine: Input a thrust table (TVCjT) for each engine and either a flowrate table (WDjT) or vacuum specific impulse (ISPV(j)) or (ISPjT)based on the value of IWDF(j).
2	Jet or ramjet engine: Input a table of net thrust (TVCjT) and specific fuel consumption (WDjT) for each engine.

NPC(9) - The propulsion type selection flag. Section 6.a.13

3	Instantaneous delta velocity addition to circularize the orbit at the current altitude or to add the desired delta velocity, DVIMAG, based on ISDVIN. Also input the specific impulse
4	Instantaneous delta velocity addition using the current weight of propellant (WPROP) and the specific impulse (ISPV(j)).

Related: NPC (30), NPC (7), NPC (27)
ASMAX - NPC (7)=1, NPC (9)=1,2 and NPC (30)=0,3,4

NPC(10) - Static trim option flag. Section 6.a.21

Value	Definition
0	**No Static trim.**
1	Static trim in pitch only.
2	Static trim in yaw only.
3	Static trim in pitch and yaw.

Related: NPC(8)

NPC(11) - The functional inequality constraints option flag. Section 6.a.7

Value	Definition
0	**No functional inequality constraints.**
1	Compute functional inequality constraints FVALi, i=1,2,3, based on the table input of the inequality boundary (FLiT, i=1,2,3).

73

NPC(12) - Crossrange and downrange option flag. Section 6.a.19

Value	Definition
0	**Do not calculate.**
1	Compute crossrange (CRRNG) and downrange (DWNRNG) based on relative great circles.
2	Compute CRRNG and DWNRNG based on inertial great circles.
3	Compute CRRNG and DWNRNG relative to the ground track of the reference circular orbit defined by ALTREF and AZREF.
4	Zero out CRRNG and compute DWNRNG using the Breguet range equation for cruise flight.

NPC(13) - The propellant jettison option flag. Section 6.a.24

Value	Definition
0	**Do not jettison the remaining propellant, WPROP.**
1	Jettison the remaining propellant, WPROP, at the beginning of the, current phase.
2	Save the amount of propellant, remaining at the end of the previous phase to be jettisoned at a later time.
3	Jettison the amount of propellant saved by NPC(13)=2 above.

Related: Used if NPC(30) = 0, NPC(17) - weight jettison option

NPC(14) - The hold-down option flag. Section 6.a.11

Value	Definition
0	**Do not use hold-down option.**
1	Integrate the equations of motion based on holding the vehicle down.
2	Use the horizontal takeoff model that maintains a constant radius plus the input radial acceleration, ZGAI.
3	Use the horizontal takeoff model that allows the user to maintain a constant altitude using altitude displacement and rate gains, ZGAG(i), i =1 or 2.

NPC(15) - Aeroheating rate option flag. Section 6.a.2

Value	Definition
0	**Do not calculate aeroheating rate.**
1	Calculate aeroheating rate (HEATRT) and total heat (TLHEAT) using Chapman equation for stagnation point heating.
2	Calculate the heating rate using HTRTT as a table look-up.
3	Calculate heating rate as the product of Options 1 and 2 above.
4	Only calculate the turbulent heating rate (HTURBD) and the turbulent heat (HTURB) using the table look-up of HTRTT as a multiplier.
5	Calculate both HEATRT and HTURBD as in the Options 3 and 4 above to yield the laminar heating (TLHEAT) and the turbulent heating (HTURB).
6	Calculate HEATRT as the maximum centerline heat rate.

Related: NPC(26), NPC(5)>0 - Atmosphere

NPC(16) - Gravity model option flag. Section 6.a.10

Value	Definition
0	**Use the gravity model for an oblate planet. Input J2, J3, J4, J5, J6, J7 J8, RE, RP, MU, and OMEGA.**
1	Use the gravity model for a spherical planet of radius RE. Input RE and MU.

NPC(17) - The weight jettison option flag based on FMASST. Section 6.a.24

Value	Definition
0	**Not used.**
1	Compute WJETTM using the mass fraction table FMASST as follows: WJETTM = WPROPI/FMASST WPROPI.
2	Set WJETT equal to WJETTM.
3	Set WJETTM equal to the table lookup of FMASST.

Related: used if NPC(30) = 0, NPC(13)

NPC(18) - A trajectory termination flag. Section 6.a.6

Value	Definition
0	**Do not terminate the trajectory.**
1	Terminate the trajectory upon reaching the current event. This option provides the user with a pseudoabort capability that can be used with roving or secondary events.

**NPC(19) - A flag to control printing of input conditions for each phase.
Section 6.a.16**

Value	Definition
0	Do not print input condition summaries.
1	**Print input condition summaries for each phase.**

**NPC(20) - A flag to specify the type of special integration step size (DT) calculation to
be used. Section 6.a.15.**

Value	Definition
0	**None.**
1	Use the argument values of the monovariant tables designated in NPC20T as integration times. The arguments for these tables must be a time parameter, e.g., TIME, TIMES, TDURP, TIMRFj, etc.
2	Use an increment in true anomaly (DTRUAN) to determine integration step size.

NPC(21) - Fuel and oxidizer weights and volumes calculations. Section 6.a.18.

Value	Definition
0	**Do not calculate fuel and oxidizer weights and volumes.**
1	Calculate fuel and oxidizer weights volumes. DENSFUEL(j), DENSOX, FUELVC(j), FUFRACT(j), OXIDVC

NPC(22) - The throttling parameter input option flag. Section 6.a.18.

Value	Definition
0	**Do not calculate the throttling parameter (ETA).**
1	The throttling parameter (ETA) is obtained by evaluating a cubic polynomial where the constant term is set equal to the value of ETA at the time NPC(22)=1 is requested. The coefficients are input as ETAPC(i),i=2,4.
2	The throttling parameter (ETA) is obtained by evaluating a cubic polynomial as when NPC(22)=1 except that the constant term is input as ETAPC(l).
3	The throttling parameter (ETA) is a table lookup of the input table ETAT.
4	The throttling parameter (ETA) is a piecewise linear function of the event specified by DESNE. Input the initial value of ETA in the first phase as ETA. The desired value of ETA at event DESNE is input as DETA.

Related: NPC (9) -NPC(22) = 1,2 and NPC(30) = 0,3,4.

NPC(23) - A flag to control velocity margin calculations. Section 6.a.26

Value	Definition
0	**Do not compute velocity margin.**
1	Input DVMARR and compute DUEXS.
2	Input DVPCT and compute DVMARR and DUEXS.
3	Input DVMARR and compute DUEXS.

NPC(24) - General integration variable flag. Section 6.a.9

Value	Definition
0	**Do not integrate special integrals.**
1	Integrate the variables specified by GDERV(i), i=1,10, to form the integrals GINTj, j=1,10.

NPC(25) - Velocity loss calculation flag. Section 6.a.25

Value	Definition
0	**Do not calculate velocity losses.**
1	Calculate the ideal velocity (VIDEAL), the drag loss (DLR or DLI), the thrust vectoring loss (TVLR or TVLI), the atmospheric thrust loss (ATL), the gravity loss (GLR or GLI), and the Coriolis loss (CLR) but do not print the velocity loss block.
2	Same as when NPC(25)=1, except print the velocity loss block only at phase changes.
3	Same as when NPC(25)=1, except print the velocity loss block at each print time.

Related: NPC (9) <> 0

NPC(26) - Special Aeroheating calculations option flag. Section 6.a.2

Value	Definition
0	**No special aeroheating calculations.**
1	Calculate aeroheating for a ten-panel vehicle model based on heating ratios referenced to the total heat (TLHEAT) calculated using NPC(15).

NPC(26) - Special Aeroheating calculations option flag. Section 6.a.2

2	Calculate special aeroheating indicators for launch vehicle ascent. These are stagnation point (AHI), dispersed stagnation point (AHIP), bottom side (HTBT), top side (HTTP), left side (HTLF), and right side (HTRT),
3	Calculate only stagnation point (AHI) and dispersed stagnation point (AHIP).
4	Calculate only AHI for a cylindrical body (Heat-Cyl) as a polynomial function of Mach number and Reynolds number.

Related: NPC(15) - AHSFT for NPC(26) = 2,3 - ARP, HRAT, WUAiT and ITAP(i) for NPC(26) = 1

NPC(27) - Activation flag for option to integrate flow rate of selected engines. Section 6.a.18

Value	Definition
0	**Inactive.**
1	Active.

Related: Used if NPC(9) = 1,2 and NPC(30) = 0, 3, 4.

NPC(28) - Tracking station option flag. Section 6.a.18

Value	Definition
0	**Do not use tracking station option.**
1	Compute tracking parameters for as many as ten tracking stations at the end of each integration step. The output variables must be requested in the print array, PRNT(i).
2	Compute tracking parameters only at phase changes and print a tracking station print block.
3	Compute tracking parameters at the end of each integration step and print a tracking station print block with each regular printout.

Related: ELEMIN(j), JTKFLG(i), TRKGLT(i), TRKLON(i), TRKNAM(i), i=1,10

NPC(29) - Analytical vacuum impact point calculation flag. Section 6.a.3

Value	Definition
0	**Do not compute impact points.**
1	Calculate impact points at the end of each integration step. The output variables must be requested in the print array, PRNT(i).
2	Compute impact points only at phase changes and print an impact point print block.
3	Compute impact points at the end of each integration step and print an impact point print block with each regular printout.

Related: ALTIP - for NPC (29) = 1,2,3

NPC(30) - A flag that specifies the vehicle weight model to be used. Section 6.a.24

Value	Definition
0	**Use the N-stage weight calculation model.**
1	Calculate weight as the sum of tables WGT1T and WGT2T as follows: WEIGHT = WGT1T + WGT2T
2	Calculate flowrate as the sum of tables WGTD1T and WGTD2T as follows: WDOT = WGTD1T + WGTD2T
3	Use the enhanced (component) weight model.
4	Same as NPC(30)=3 except that WPRP(i) is obtained as a table lookup of WPRPiT,i=1,15. NPC(30)=3 should be input in the next phase.

Related:

IEGMF(J) - on/off and engine type selection - NPC(30) = 3 or 4
ISTEPF(j) - Activation/Dry weight model - NPC(30) = 3 or 4
IWJF(j) - Engine (j) Propellant Jettison - NPC(30) = 3 or 4
IWSDF(i) - Dry rate flowrate activation flag - NPC(30) = 3 or 4
MENSTP(i)
NENGH - Highest index of any engine currently on vehicle - NPC(30) = 3 or 4

81

NENGL - Lowest index of any engine currently on vehicle - NPC(30) = 3 or 4

NSTPH - Highest index of any step currently on vehicle - NPC(30) = 3 or 4

NSTPL - Lowest index of any step currently on vehicle - NPC(30) = 3 or 4

PWPROP - Amount propellant consumed by engine IWPF(i) - NPC(27) = 1 and NPC(30) = 0,3,4

WEICON - Initial value consumed

WGSTG - Vehicle gross weight at phase entered - uses NPC(13) and NPC(17) and used in NPC(30) = 0

WJETT - Weight to be jettisoned at beginning of phase WJETT is input - NPC(30) = 0

WPLD - Payload weight - only inputted if WGTSG is also inputted - NPC(30) = 0

WPROPI - initial propellant weight - NPC(30) = 3 or 4

WPRN(j) - Current nonusable propellant weight for engine (j) - NPC(30) = 3 or 4

WPRP(j) - Current total propellant weight for engine (j) -- NPC(30) = 3 or 4

WSJTD(i) - Current dry weight to be jettisoned from vehicle step (i) - NPC(30) = 3 or 4

WSJTP(j) - Current propellant weight to be jettisoned from engine (j) - NPC(30) = 3 or 4

WSTPD(i) - Dry weight for vehicle step (i), used if ISTEPF(i) = 1 and NPC(30) = 3 or 4

NPC(31) - A flag to activate the vernal equinox, sun shadow, and sun angle option. Section 6.a.27

Value	Definition
-1	Force the ECI frame to coincide with vernal equinox frame (the program calculates TRPM).
0	**Do not activate this option.**
1	Activate this option.

Related: Input - Date(i) i=1,3 (month, day, year)

DECL, GAA, GHAS, - used if date (i) not used

TRPM - NPC(31) = 1

TSIPM - input in first phase

NPC(32) - The parachute drag option flag. Section 6.a.28

Value	Definition
0	**Do not compute parachute drag.**
1	Compute parachute drag with VELAP=VELA at the beginning of the current phase.
2	Compute parachute drag with VELAP= current value of VELA.

NPC(33) - A flag to activate the calculation of the radio guidance (BTL) coordinates Section 6.a.33

Value	Definition
0	**Do not compute BTL coordinates.**
>0	Calculate BTL coordinates using the tracker designated by the value of NPC(33)>1 and <=10.

NPC(34) - The desired partial Keplerian state specification option flag. Section 6.a.31

Value	Definition
0	**Do not perform calculations.**
1	Perform calculations with SMJAXD and ECCEND used to define orbit size and shape.
2	Perform calculations with PGERDD and APORDD used to define orbit size and shape.

NPC(35) - A flag to activate the integration of sensed velocity increment. Section 6.a.33

Value	Definition
0	**Do not activate option.**
1	Activate option.

Related: IARCP - Activate ARC length calculations along velocity vector.

NPC(36) - Sunlight option flag. Section 6.a.32

Value	Definition
0	**Do not activate option.**
1	Activate option.
2	Activate option and print sunlight print block.

NPC(37) - Date option fag. Section 6.a.32

Value	Definition
0	**Maintain date at date 1 for entire simulation.**
1	Increment date with simulation.

NPC(38) - Atmospheric turbulence flag. Section 6.a.4

Value	Definition
0	**No turbulence.**
1	Dryden turbulence with initialization.
2	Dryden turbulence but do not initialize random number generator. The model will set NPC(38) after one pass with NPC(38)=1.

Related: NPC(6) <> 0

NPC(39) - Atmospheric turbulence flag. Section 6.a.4

Value	Definition
0	**No Gusts.**
1	User defines gusts with VTFREQ(j) and VTMAG(j), j=1,3.

Related: NPC(6) <> 0

IGUID CODES

IGUID(1) - Type of guidance (steering) desired. Section 6.b-1

Value	Definition
-1	Inertial body rate polynomials.
0	**Angle of attack, sideslip, and bank.**
1	Inertial Euler attitude angles, i.e., ROLI, YAWI, and PITI measured with respect to the L frame.
2	Relative Euler attitude angles, i.e., YAWR, PITR, and ROLR measured with respect to the G-frame.

85

IGUID(1) - Type of guidance (steering) desired. Section 6.b-1

3	Aerodynamic angles with respect to the inertial velocity vector ALPHI, BETAI, and BANKI.
4	Pitch-plane angles, i.e., ROLI, YAWI, and PITI.

IGUID(2) - Attitude channel selector. Section 6.b

Value	Definition
0	**Calculate all attitude channels based upon the same type of functional relationship, i.e., polynomials, tables, etc.**
1	Calculate each attitude channel separately by a functional relationship specified by IGUID(6), (7) and (8) or IGUID(9), (10) and (11). This flag enables one to select different types of aerodynamic angle steering in each attitude channel

IGUID(3) - Steering option to command all channels simultaneously. Section 6.b

Value	Definition
0	**Command angle of attack, sideslip, and bank as third order polynomials with the values of ALPHA, BETA, and BNKANG carried over from the previous phase.**
1	The same as IGUID(3)=0 except that the constant terms of the polynomials are the input values. This generally causes an instantaneous change in attitude at the beginning of the phase.
2	Angle of attack, sideslip, and bank are obtained from table lookup of ALPHAT, BETAT, and BANKT.
3	Angle of attack, sideslip, and bank are piecewise linear functions of the CRITR variable at the events DESN(i), i=1,2,3, respectively
4	AOA, sideslip, and bank are computed via linear feedback to make the variable specified by DGF(i), i=1,2,3 follow the profile contained in GDFiT, i=1,2,3 for angle of attack, sideslip, and bank, respectively.

IGUID(3) - Steering option to command all channels simultaneously. Section 6.b

5	The same as IGUID(3)=0 except that the constant terms are the desired incremental values of ALPHA, BETA, and BNKANG at the beginning of the phase; e.g., the internal value of ALPPC(1) = ALPHA + the input value of ALPPC(1).

IGUID(4) - Euler angle steering (inertial or relative). Section 6.b

Value	Definition
0	**YAWR, PITR, and ROLR are computed as third-order polynomials with the values of YAWR, PITR, and ROLR carried over from the previous phase. That is, the constant terms of the polynomials are set equal to the values of YAWR, PITR, and ROLR at the beginning of the phase.**
1	YAWR, PITR, and ROLR are given by third-order polynomials as in IGUID(4)=0, except that the constant terms of the polynomials are the input values. This generally causes an instantaneous change in attitude at the beginning of the phase.
2	YAWR, PITR, and ROLR are computed from tables of YAWT, PITT, and ROLT.
3	YAWR, PITR, and ROLR are piecewise linear functions of the CRITR variable at the events DESN(1), DESN(2), and DESN(3), respectively.
4	YAWR, PITR, and ROLR are computed via linear feedback to make the variable specified by DGF(i), i=1,2,3 follow the profile contained in GDFiT, i=1,2,3 for YAWR, PITR, and ROLR, respectively.
5	Same as IGUID(4)=0 except that the constant terms are the desired incremental values of YAWR, PITR, and ROLR at the beginning of the phase; e.g., the internal value of PITPC(1) = PITR plus the input value of PITPC(1).
6	ROLLPC(2),PITPC(2), and YAWPC(2) are computed from table lookups of ROLT, PITT, and YAWT.

IGUID(5) - A flag to determine the method of calculating the body rates. Section 6.b

Value	Definition
1	**ROLBD, PITBD, and YAWBD polynomials.**
2	BNKDOT, PITBD, and YAWBD polynomials.
3	ROLBD, ALPDOT, and YAWBD polynomials.
4	ROLBD, PITBD, and BETDOT polynomials.
5	BNKDOT, ALPDOT, and YAWBD polynomials.
6	ROLBD, ALPDOT, and BETDOT polynomials.
7	BNKDOT, PITBD, and BETDOT polynomials.
8	ALPDOT, BETDOT, and BNKDOT polynomials.
9	YAWRD, PITRD, and ROLRD polynomials.
10	ROLBD and YAWBD polynomials with ALPDOT computed to drive ALPHA from its current value to the value input as DALPHA at the beginning of the next primary phase. This allows the user to drive ALPHA to a desired value while staying in the same inertial pitch plane.
13	Calculate the constant body rate magnitude OMGBD based on the CRITR value of DESN, which must be a time duration, e.g. TDURP.
14	Calculate the CRITR value of DESN based on the input value of OMGBD.

IGUID(6) - Separate channel options for angle of attack. Section 6.b

Value	Definition
0	**Same as when IGUID(3)=0.**
1	Same as when IGUID(3)=1.
2	Same as when IGUID(3)=2.

IGUID(6) - Separate channel options for angle of attack. Section 6.b

3	Same as when IGUID(3)=3.
4	Same as when IGUID(3)=4.
5	Same as when IGUID(3)=5.

IGUID(7) - Separate channel options for sideslip angle. Section 6.b

Value	Definition
0	**Same as when IGUID(3)=0.**
1	Same as when IGUID(3)=1.
2	Same as when IGUID(3)=2.
3	Same as when IGUID(3)=3.
4	Same as when IGUID(3)=4.
5	Same as when IGUID(3)=5.

IGUID(8) - Separate channel options for bank angle. Section 6.b

Value	Definition
0	**Same as when IGUID(3)=0.**
1	Same as when IGUID(3)=1.
2	Same as when IGUID(3)=2.
3	Same as when IGUID(3)=3.
4	Same as when IGUID(3)=4.
5	Same as when IGUID(3)=5.

IGUID(9) - Separate channel options for YAWR angle. Section 6.b

Value	Definition
0	**Same as when IGUID(4)=0.**
1	Same as when IGUID(4)=1.
2	Same as when IGUID(4)=2.
3	Same as when IGUID(4)=3.
4	Same as when IGUID(4)=4.
5	Same as when IGUID(4)=5.

IGUID(10) - Separate channel options for PITR angle. Section 6.b

Value	Definition
0	**Same as when IGUID(4)=0.**
1	Same as when IGUID(4)=1.
2	Same as when IGUID(4)=2.
3	Same as when IGUID(4)=3.
4	Same as when IGUID(4)=4.
5	Same as when IGUID(4)=5.

IGUID(11) - Separate channel options for ROLR angle. Section 6.b

Value	Definition
0	**Same as when IGUID(4)=0.**
1	Same as when IGUID(4)=1.
2	Same as when IGUID(4)=2.

IGUID(11) - Separate channel options for ROLR angle. Section 6.b

3	Same as when IGUID(4)=3.
4	Same as when IGUID(4)=4.
5	Same as when IGUID(4)=5.

IGUID(12) - Inertial body rate initialization flag. Section 6.b

Value	Definition
1	Initialize body rates using ALPHA, BETA, and BNKANG.
2	**Initialize body rates using ROLI, YAWI, and PITI.**

IGUID(13) - The YAWR angle reference option. Section 6.b

Value	Definition
1	**Relative yaw angle (YAWR) is measured clockwise from geographic north.**
2	Relative yaw angle (YAWR) is measured clockwise from the atmospheric relative velocity vector azimuth angle.
3	Relative yaw angle (YAWR) is measured clockwise from the inertial velocity vector azimuth angle.

IGUID(14) - The general open/closed-loop guidance option selection flag. Section 6.b

Value	Definition
0	**Do not use the general open-loop guidance option.**
1	Use the general open-loop guidance option.

IGUID(14) - The general open/closed-loop guidance option selection flag. Section 6.b

2	Use the general closed-loop guidance programmed in subroutine CLGM.
3	Use open-loop approximation to linear sine steering.

IGUID(15) - The general open-loop guidance override selection flag. Section 6.b

Value	Definition
0	**Do not use.**
1	Use the general open-loop guidance override model in subroutine OLGOM.

IGUID(16) - The pitch-plane angle selection flag. Section 6.b

Value	Definition
0	**PITI = the angle from the L-frame x axis after inertial Euler ROLI and YAWI rotations to body roll (XB) axis. This is equivalent to IGUID(1)=1.**
1	PITI = the angle from the atmospheric velocity vector to the body roll (XB) axis.
2	PITI = the angle from the inertial velocity vector to the body roll (XB) axis.
3	PITI = the angle of the body roll (XB) axis above the local horizontal plane.

APPENDIX B. EXAMPLE PROBLEM

This appendix contains Example Problem 1 from the POST utilization Manual [Ref 1].

EXAMPLE PROBLEM 1

An important ascent trajectory optimization problem during the conceptual phases of the Space Shuttle program was that of determining the maximum payload capability of various configuration concepts. One such Space Shuttle configuration is represented by this sample problem. The four key components of this configuration are the orbiter, two solid rocket boosters (SRBs), and the external tank (ET). This multibody nonsymmetrical configuration created special simulation requirements that motivated many of the features contained in POST. For example, to accurately predict the performance capability of a unsymmetrical configuration such as Space Shuttle, it is important to include the thrust vectoring losses encountered as the engines gimbal to balance the aerodynamic moments caused by the nonsymmetrical shape of the configuration. This fact led to the development of the static trim option employed in this sample case.

Problem Formulation

There are a number of ways to formulate the problem of maximizing payload for a given configuration. Each approach is based on (1) what is known about the configuration, and (2) what is known about the basic trajectory to be flown. In this first example, it is assumed that the user knows the dry weight and the propellant weight of each of the four major components of the vehicle. Assuming that all the propellant is consumed during the flight, which is ensured by terminating the simulation on the event criteria

$$Wprop = 0,$$

enables the payload weight to be computed from the equation

$$W_{PLD} = W_{BO} - W_{dry},$$

where W_{BO} is the total burnout weight (at the final event) and W_{dry} is the known weight of the remaining vehicle components. Because W_{dry} is constant for a given configuration,

93

maximizing WW_{BO} is equivalent to maximizing W_{dry}. Thus, in this example, the optimization variable was selected to be WW_{BO}, which is computed as the weight of the vehicle, WEIGHT, at the instant that the weight of propellant is zero.

Trajectory Profile

As in any trajectory problem, there are a variety of ways in which to simulate this mission, and the following sequence of events illustrates the approach taken in this example:

Trajectory Profile For Sample Problem 1

Event Number	Description
1.0	Lift-off at t = 0s.
2.0	Interrupt at t = 15s to initiate Pitch Rate 1.
3.0	Interrupt at t = 25s to initiate Pitch Rate 2.
4.0	Interrupt at t = 40s to initiate Pitch Rate 3.
5.0	Interrupt at t = 60s to initiate Pitch Rate 4.
6.0	Interrupt at t = 120s to initiate Pitch Rate 5.
7.0	Interrupt at t = 150s to initiate Pitch Rate 6.
8.0	Interrupt when the remaining propellant (WPROP) equals zero to initiate a coast phase.
9.0	Interrupt at 7s from the phase 8.0 to initiate jettison of Stage 1 and start of Stage 2 flight. Also initiate Pitch Rate 7
10.0	Interrupt at 100s from phase 9.0 to initiate Pitch Rate 8
11.0	Interrupt at 150s from phase 10.0 to activate conic calculations
12.0	Terminate when the remaining propellant (WPROP) equals zero.

As indicated, the simulation starts with a 15-second vertical rise, followed by a sequence of constant pitch rate steering segments. The static trim option is used during all early flight phases, and a 3-g acceleration limit is enforced after 60 seconds of flight. Event

8.0 specifies burnout of the SRBs, which are jettisoned seven seconds later at Event 9.0. Notice also that new data for propulsion and aerodynamics are input in Event 9.0. These data represent the orbiter plus the external tank combination that is used for the remainder of the trajectory simulation. As mentioned earlier, the final event criterion is the weight of propellant. Because the last initialization of weight of propellant was in Event 9.0, the program variable Wprop represents the amount of propellant in the orbiter plus the external tank combination at any time. Thus, the final condition

$$Wprop = 0,$$

limits the amount of propellant that can be consumed in all flight phases after the occurrence of Event 9.0.

Targeting and Optimization Formulation

In this example, the mission requirements are the delivery of the payload to the perigee of a 50x100 nmi parking orbit. These requirements are mathematically equivalent to the three terminal equality constraints

$$h_f = 303\ 805.0 \text{ ft.}$$

$$V_{If} = 25\ 853.0 \text{ fps}$$

$$\gamma_{If} = 0.0 \text{ deg}$$

where the subscript, f, denotes final burnout conditions. Extensive computational experience indicates that the spherical-coordinate constraints (h, VI, γ_I) are easier to satisfy than their orbital counterparts (hp, ha, θ). The reason for this is probably related to the nonlinearities involved, with (h, VI, γ_I) appearing more linear in the independent variables.

Finally, the control parameters selected are the gross vehicle weight at lift-off and the eight inertial Eulerian pitch rates throughout the trajectory. Six of these pitch rates are used to steer the vehicle during the SRB boost phases and two during the exoatmospheric phases. The motivation for using these particular control variables is computational experience, which shows that pitch angle steering is an efficient technique

for optimizing ascent trajectories. The particular "break times" in the pitch history were selected after a few single-pass simulations. It is generally not a good practice to allow the program to decide on both the "break times" and the rates, although it might seem logical to do so. The initial gross weight of the vehicle, W_G, is employed as an independent variable to maximize the payload because in this setup there is a direct one-to-one correspondence between an increment in W_{PLD} and an increment in W_G because all vehicle dry weights and propellant weights are held constant during the optimization.

The previous discussion can be summarized by stating the precise mathematical formulation of the problem: Determine the control parameters

$\underline{u} = (W_G, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8),$

that maximize: W_{BO}

subject to:

$h_f = 303\ 805.0$ ft.

$V_{If} = 25\ 853.0$ fps

$\gamma_{If} = 0.0$ deg

APPENDIX C. INPUT DATA FILE FOR EXAMPLE PROBLEM 1

This appendix contains the data file corresponding to the problem definition in Appendix B.

```
l$search
c******************************************************************
c problem
c    maximize weight
c    subject to
c            altito - 303805  = 0
c            veli   -  25853  = 0
c            gammai -    0    = 0
c******************************************************************
c
c
c listin = 1,
 maxitr  =      10,
 srchm   =       4,
 opt     =     1.0,
 optvar  = 6hweight,
 optph   =    12.0,
 wopt    =  1.0e-6,
c
 nindv   =       9,
 pert    =     1.0,
 indvr   = 6hwgtsg , 6hpitpc2, 6hpitpc2, 6hpitpc2, 6hpitpc2,
 indph   =       1,      2,      3,      4,      5,
 u       =4031000.0,   -1.8,    -.5,     -.2,    -.3,
 indvr(6) = 6hpitpc2, 6hpitpc2, 6hpitpc2, 6hpitpc2,
 indph(6) =       6,      7,      9,     10,
 u(6)    =    -.25,    -.3,    -.15,   -.05,
c
 ndepv   =       3,
 depvr   = 6haltito, 6hveli  , 6hgammai,
 depval  = 303805.0, 25853.0,    0.0,
 deptl   =   100.0,     .1,    .001,
 $
l$gendat
prnc=0,
prnca=0,
title  = 0h* sample problem for ascent trajectory w/ drop tank orbiter*,
event   =       1,
```

97

```
c
npc(2)   =     1, 4, 2,
npc(8)   =     2, 1,
npc(16)  =     1,
npc(21)  =     1,
iwdf(1) = 2,
c
iguid(1) =     1,
iguid(4) =     1,
c
maxtim  =  1000.0,
altmax  =2000000.0,
fesn    =     12,
dt      =    5.0,
pinc    =    20.0,
gdlat   =    28.5,
long    =   279.4,
azl     =    90.0,
neng    =     1,
wpropi  =2249000.0,
ispv    =   439.0,
gxp     =   218.42,
gyp     =     0.0,
gzp     =    33.33,
sref    =  4500.00,
lref    =  218.833,
$
l$tblmlt
$
l$tab
table    = 6htvc1t ,0,5472000.0,
$
l$tab
table    = 6hae1t ,0,232.5,
$
l$tab
table    = 6hcdt  ,2,6hmach ,6halpha ,12,5,1,1,1,1,1,1,1,1,
-20.,
0.0, 1.456,  .5, 1.585, .7, 1.598, .8, 1.242, 1., 3.157, 1.2,2.996,
1.5, 1.816, 2.0, 1.301, 3.,  .850, 5.,  .482, 7.,  .382, 10.,  .396,
- 5.,
0.0,  .263,  .5,  .338, .7,  .110, .8,  .302, 1.,  .690, 1.2, .671,
1.5,  .563, 2.0,  .480, 3.,  .383, 5.,  .256, 7.,  .212, 10.,  .210,
```

```
     0.,
0.0,  .180,  .5,  .18 , .7,  .200, .8,  .251, 1.,  .495, 1.2, .502,
1.5,  .485, 2.0,  .456, 3.,  .391, 5.,  .272, 7.,  .231, 10., .231,
  5.,
0.0,  .263,  .5,  .338, .7,  .110, .8,  .302, 1.,  .690, 1.2, .671,
1.5,  .563, 2.0,  .480, 3.,  .383, 5.,  .256, 7.,  .212, 10., .210,
 20.,
0.0, 1.456,  .5, 1.585, .7, 1.598, .8, 1.242, 1., 3.157, 1.2,2.996,
1.5, 1.816, 2.0, 1.301, 3.,  .850, 5.,  .482, 7.,  .382, 10., .396,
 $
l$tab
table   = 6hclt  ,2,6hmach ,6halpha ,12,4,1,1,1,1,1,1,1,1,
-20.,
0.0, -1.010,  .5,-1.025, .7, -.99,  .8, -.815, 1.,-1.08 , 1.2,-1.11,
1.5,  -.895, 2.0, -.788, 3.,-.635,  5., -.480, 7., -.43 , 10., -.43,
  0.,
0.0,  .015,  .5,  .04 , .7, .01 ,  .8, -.045, 1.,  .08 , 1.2, .038,
1.5,  -.02 , 2.0, -.108, 3.,-.145,  5., -.15,  7., -.15 , 10.,-.15 ,
  5.,
0.0,  .545,  .5,  .75, .7,  .53,  .8,  .365, 1.,  .69 , 1.2, .638,
1.5,   .43 , 2. ,  .242, 3.,  .11,  5., .025 , 7.,  .00 , 10., .00 ,
 20.,
0.0, 2.135,  .5, 2.24 , .7, 2.09,  .8,1.595 , 1., 2.52 , 1.2,2.438,
1.5,  1.78 , 2. , 1.292, 3., .875,  5., .55  , 7.,  .45 , 10., .45 ,
 $
l$tab
table   = 6hcmat ,1,6hmach ,12,1,1,1,
0.0, .019, .7, .0218, .9, .0302, 1., .023, 1.2,-.011, 1.5,-.032,
1.8,-.0395, 2.,-.0419, 3.,-.0396, 5.,-.0187, 7.,-.0082, 10., 0.0,
 $
l$tab
table   = 6hxreft ,1,6hmach ,12,1,1,1,
0.0,137.86, .7,140.05, .9,136.77, 1.,147.71, 1.2,145.52,
1.5,144.43, 1.8,141.58, 2.,138.3, 3.,131.74, 5.,118.83,
7.,109.42, 10.,91.91,
endphs  =      1,
 $
l$gendat
event   =      2, critr   = 6htime , value   = 15.0,
iguid(4) =      0,
endphs  =      1,
 $
l$gendat
```

```
event   =       3, critr   = 6htime , value    = 25.0,
endphs  =       1,
$
l$gendat
event   =       4, critr   = 6htime , value    = 40.0,
endphs  =       1,
$
l$gendat
event   =       5, critr   = 6htime , value    = 60.0,
endphs  =       1,
$
l$gendat
event   =       6, critr   = 6htime , value    = 120.0,
endphs  =       1,
$
l$gendat
event   =       7, critr   = 6htime , value    = 150.0,
dt      =      10.0,
endphs  =       1,
$
l$gendat
event   =       8, critr   = 6hwprop , value   = 0.0,
tol     =      2.e-6,
npc(9)  =       0, 0,
weicon  =      0.0,
endphs  =       1,
$
l$gendat
event   =       9, critr   = 6htdurp , value   = 7.,
tol     =      1.e-6,
dt      =      20.0,
pinc    =      50.0,
npc(9)  =       1,
wjett   = 665000.0,
wpropi  = 809000.0,
ispv    =     459.0,
gxp     =     142.0,
gyp     =      0.0,
gzp     =      25.0,
sref    =    4840.0,
lref    =     135.0,
$
```

l$tblmlt
 $
l$tab
 table   = 6htvclt ,0,1431000.0,
 $
l$tab
 table   = 6hae1t ,0,154.54,
 $
l$tab
 table = 6hcdt   ,2,6hmach ,6halpha ,12,7,1,1,1,1,1,1,1,1,
 -20.,
 0,.024, .2,.024, .6,.026, .8,.028, .9,.035, 1.3,.093, 1.5,.122,
 2,.116, 2.48,.1, 3,.092, 3.9,.082, 40,.03,
 - 4.,
 0,.024, .2,.024, .6,.026, .8,.028, .9,.035, 1.3,.093, 1.5,.122,
 2,.116, 2.48,.1, 3,.092, 3.9,.082, 40,.03,
  0.,
 0,.026, .2,.026, .6,.026, .8,.024, .9,.036, 1.3,.092, 1.5,.118,
 2,.106, 2.48,.091, 3,.082, 3.9,.074, 40,.022,
  5.,
 0,.042, .2,.042, .6,.04, .8,.042, .9,.076, 1.3,.124, 1.5,.142,
 2,.124, 2.48,.098, 3,.088, 3.9,.079, 40,.033,
  10.,
 0,.076, .2,.076, .6,.08, .8,.1, .9,.13, 1.3,.194, 1.5,.192,
 2,.165, 2.48,.127, 3,.114, 3.9,.095, 40,.057,
  20.,
 0,.36, .2,.36, .6,.362, .8,.44, .9,.41, 1.3,.39, 1.5,.36 , 2,.32,
 2.48,.242, 3,.224, 3.9,.216, 40,.238,
  30.,
 0,.36, .2,.36, .6,.36, .8,.44, .9,.41, 1.3,.39, 1.5,.36, 2,.32,
 2.48,.44, 3,.418, 3.9,.4, 40,.3,
 $
l$tab
 table = 6hclt   ,2,6hmach ,6halpha ,12,7,1,1,1,1,1,1,1,1,
 -20.,
 0,-.07, .2,-.08, .6,-.12, .8,-.12, .9,-.12, 1.3,-.12, 1.5,-.12,
 2,-.13, 2.48,-.14, 3,-.12, 3.9,-.1, 40,-.14,
  4.,
 0,-.07, .2,-.08, .6,-.12, .8,-.12, .9,-.12, 1.3,-.12, 1.5,-.12,
 2,-.13, 2.48,-.14, 3,-.12, 3.9,-.1, 40,-.14,
  0.,
 0,.08, .2,.08, .6,.08, .8,.06, .9,.06, 1.3,.07, 1.5,.04, 2,0.0,
 2.48,-.02, 3,-.03, 3.9,-.04, 40,.03,

```
    5.,
0,.29, .2,.29, .6,.29, .8,.28, .9,.28, 1.3,.3, 1.5,.24, 2,.17,
2.48,.12, 3,.09, 3.9,.08, 40,.21,
 10.,
0,.5, .2,.6, .6,.49, .8,.48, .9,.52, 1.3,.52, 1.5,.41, 2,.33,
2.48,.25, 3,.2, 3.9,.15, 40,.4,
 20.,
0,.94, .2,.94, .6,.92, .8,.9, .9,.94, 1.3,.89, 1.5,.75, 2,.63,
2.48,.51, 3,.43, 3.9,.39, 40,.76,
 30.,
0,.94, .2,.94, .6,.92, .8,.9, .9,.94, 1.3,.89, 1.5,.75, 2,.68,
2.48,.67, 3,.65, 3.9,.62, 40,.76,
$
l$tab
table    = 6hcmat  ,0,0.0,
$
l$tab
table    = 6hxcgt  ,1,6hweicon,5,1,1,1,
0,87.64, 202250,93.93, 404500,99.68, 606750,104.04, 809000,104.37,
$
l$tab
table    = 6hycgt  ,0,0.0,
$
l$tab
table    = 6hzcgt  ,1,6hweicon,5,1,1,1,
0,31.33, 202250,31.5, 404500,31.75, 606750,32.42, 809000,33.83,
endphs  =       1,
$
l$gendat
event   =       10, critr   = 6htdurp , value    = 100.0,
endphs  =       1,
$
l$gendat
event   =       11, critr   = 6htdurp , value    = 150.0,
npc(1)  =       2,
endphs  =       1,
$
l$gendat
event   =       12, critr   = 6hwprop , value    = 0.,
endphs  =       1,
endprb  =       1,
endjob  =       1,
```

APPENDIX D. INPUT DATA FILE GENERATED BY G-POST

This appendix contains the data file **tutorial.inp**. This file is the G-POST generated input file corresponding to the problem definition in Appendix B.

```
l$SEARCH
SRCHM   =  4,           / Optimization technique
MAXITR  =  10,          / Maximum number of itterations
OPT     =  1.0,         / Maximization
OPTVAR  =  6HWEIGHT,    / The optimization variable
OPTPH   =  12.0,        / The optimization phase
WOPT    =  1.0E-6,      / The optimization weighting
C
NINDV   =  9,
INDVR   =  6HWGTSG ,6HPITPC2,6HPITPC2,6HPITPC2,6HPITPC2,
INDPH   =  1,2,3,4,5,      / Independant Variable phases
U       =  4031000.,-1.8,-.5,-.2,-.3,      / Independant variable initial guesses
PERT    =  1.0,1.0,1.0,1.0,1.0,      / Independant variable perturbations
C
INDVR(6) =  6HPITPC2,6HPITPC2,6HPITPC2,6HPITPC2,
INDPH(6) =  6,7,9,10,      / Independant Variable phases
U(6)     =  -.25,-.3,-.15,-.05,      / Independant variable initial guesses
PERT    =  1.0,1.0,1.0,1.0,      / Independant variable perturbations
C
NDEPV   =  3,
DEPVR   =  6HALTITO,6HVELI,6HGAMMAI, / Dependant Variable names
DEPVAL  =  303805.0,25853.0,0.0,      / Dependant Variable values
DEPTL   =  100.0,.1,.001,      / Dependant variable tolerances
$
l$GENDAT
PRNC    =  0,           / Profil binary print interval
PRNCA   =  0,           / Profil ASCII print interval
PINC    =  20,          / Print interval
TITLE   =  0H*sample problem for ascent trajectory w/ drop tank orbiter*,
EVENT   =  1,
NPC(2)  =  1,           / Runge-Kutta integration
DT      =  5.0,         / Step size
EPSINT  =  0.01,        / Error tolerance
NPC(5)  =  2,           / 1962 US atmosphere
NPC(26) =  0,           / Aeroheating flag.
NPC(16) =  1,           / Oblate planet gravity model
```

```
NPC(3)   =  4,           / Velocity vector
NPC(4)   =  2,           / Spherical position vector
AZL      =  0,           / Azimuth
GDLAT    =  0,           / Geodetic latitude
LONG     =  0,           / Relative longitude
NPC(8)   =  2,           / Drag and lift coefficients
GXP      =  218.42,      / X location of engine gimbal
GYP      =  0.0,         / Y location of engine gimbal
GZP      =  33.33,       / Z location of engine gimbal
SREF     =  4500.00,     / Reference area
LREF     =  218.833,     / Reference length
NPC(9)   =  1,           / Rocket propulsion
NENG     =  1,           / Number of engines
IWDF(1)  =  2,           / Engine flowrate
ISPV     =  439.0,       / Engine impulse
WPROPI   =  2249000.0,   / Initial propellant weight
NPC(21)  =  1,           / Calculate propellant weights
IGUID(1) =  1,           / Inertial Euler angles guidance
IGUID(4) =  1,           / Polynomials w/input values
MAXTIM   =  1000.0,      / Maximum time before termination
ALTMAX   =  2000000.0,   / Maximum altitude before termination
FESN     =  12,          / Final event number
$
l$TBLMLT
$
l$TAB
TABLE    =  6htvc1t ,0,5472000.0,
$
l$TAB
TABLE    =  6hae1t ,0,232.5,
$
l$TAB
TABLE    =  6hcdt  ,2,6hmach ,6halpha ,12,5,1,1,1,1,1,1,1,1,
-20.,
 0.0, 1.456, .5, 1.585, .7, 1.598, .8, 1.242, 1., 3.157, 1.2,2.996,
1.5, 1.816, 2.0, 1.301, 3., .850, 5., .482, 7., .382, 10., .396,
- 5.,
0.0, .263, .5, .338, .7,  .110, .8,  .302, 1.,  .690, 1.2, .671,
1.5, .563, 2.0, .480, 3., .383, 5., .256, 7., .212, 10., .210,
 0.,
0.0, .180, .5, .18 , .7, .200, .8, .251, 1., .495, 1.2, .502,
1.5, .485, 2.0, .456, 3., .391, 5., .272, 7., .231, 10., .231,
 5.,
```

```
  0.0,  .263,  .5,  .338, .7,  .110, .8,  .302, 1.,  .690, 1.2, .671,
  1.5,  .563, 2.0,  .480, 3.,  .383, 5.,  .256, 7.,  .212, 10., .210,
   20.,
  0.0, 1.456,  .5, 1.585, .7, 1.598, .8, 1.242, 1., 3.157, 1.2,2.996,
  1.5, 1.816, 2.0, 1.301, 3.,  .850, 5.,  .482, 7.,  .382, 10., .396,
  $
l$TAB
  TABLE   = 6hclt  ,2,6hmach ,6halpha ,12,4,1,1,1,1,1,1,1,1,
  -20.,
  0.0, -1.010,  .5,-1.025, .7, -.99,  .8, -.815, 1.,-1.08 , 1.2,-1.11,
  1.5,  -.895, 2.0, -.788, 3.,-.635,  5., -.480, 7., -.43 , 10., -.43,
   0.,
  0.0,  .015,  .5,  .04 , .7, .01 ,  .8, -.045, 1.,  .08 , 1.2, .038,
  1.5,  -.02 , 2.0, -.108, 3.,-.145,  5., -.15,  7., -.15 , 10.,-.15 ,
   5.,
  0.0,  .545,  .5,  .75, .7,  .53,  .8,  .365, 1.,  .69 , 1.2, .638,
  1.5,  .43 , 2. ,  .242, 3.,  .11,  5., .025 , 7.,  .00 , 10., .00 ,
   20.,
  0.0, 2.135,  .5, 2.24 , .7, 2.09,  .8,1.595 , 1., 2.52 , 1.2,2.438,
  1.5,  1.78 , 2. , 1.292, 3., .875, 5.,  .55 , 7.,  .45 , 10., .45 ,
  $
l$TAB
  TABLE   = 6hcmat ,1,6hmach ,12,1,1,1,
  0.0, .019, .7, .0218, .9, .0302, 1., .023, 1.2,-.011, 1.5,-.032,
  1.8,-.0395, 2.,-.0419, 3.,-.0396, 5.,-.0187, 7.,-.0082, 10., 0.0,
  $
l$TAB
  TABLE   = 6hxreft ,1,6hmach ,12,1,1,1,
  0.0,137.86, .7,140.05, .9,136.77, 1.,147.71, 1.2,145.52,
  1.5,144.43, 1.8,141.58, 2.,138.3, 3.,131.74, 5.,118.83,
  7.,109.42, 10.,91.91,
  ENDPHS  =      1,
  $
l$GENDAT
  EVENT   =      2,
  CRITR   =      6htime ,
  VALUE   =     15.0,
  IGUID(4) =     0,            / Polynomial w/constant term carried over
  ENDPHS  =      1,
  $
l$GENDAT
  EVENT   =      3,
  CRITR   =      6htime ,
```

```
 VALUE   =    25.0,
 ENDPHS  =     1,
 $
1$GENDAT
 EVENT   =     4,
 CRITR   =    6htime ,
 VALUE   =    40.0,
 ENDPHS  =     1,
 $
1$GENDAT
 EVENT   =     5,
 CRITR   =    6htime ,
 VALUE   =    60.0,
 ENDPHS  =     1,
 $
1$GENDAT
 EVENT   =     6,
 CRITR   =    6htime ,
 VALUE   =    120.0,
 ENDPHS  =     1,
 $
1$GENDAT
 EVENT   =     7,
 CRITR   =    6htime ,
 VALUE   =    150.0,
 DT      =    10.0,          / Step size
 ENDPHS  =     1,
 $
1$GENDAT
 EVENT   =     8,
 CRITR   =    6hwprop ,
 VALUE   =     0.0,
 TOL     =    2.e-6,         / Accuracy of CRITR
 NPC(9)  =     0, 0,         / No thrust
 WEICON  =     0.0,          / Propellant consumed
 ENDPHS  =     1,
 $
1$GENDAT
 EVENT   =     9,
 CRITR   =    6htdurp ,
 VALUE   = 7.,
 TOL     =    1.e-6,         / Accuracy of CRITR
 DT      =    20.0,          / Step size
```

```
     PINC    =    50.0,            / Print interval
     NPC(9)  =     1,              / Rocket propulsion
     WJETT   =    665000.0,            / Weight to be jettisoned
     WPROPI  =     809000.0,              / Initial propellant weight
     ISPV    =     459.0,          / Engine impulse
     GXP     =    142.0,           / X location of engine gimbal
     GYP     =     0.0,            / Y location of engine gimbal
     GZP     =     25.0,           / Z location of engine gimbaL
     SREF    =    4840.0,              / Reference area
     LREF    =    135.0,              / Reference length
     $
1$TBLMLT
     $
1$TAB
     TABLE   = 6htvc1t ,0,1431000.0,
     $
1$TAB
     TABLE   = 6hae1t ,0,154.54,
     $
1$TAB
     TABLE   = 6hcdt   ,2,6hmach  ,6halpha ,12,7,1,1,1,1,1,1,1,1,1,
     -20.,
     0,.024, .2,.024, .6,.026, .8,.028, .9,.035, 1.3,.093, 1.5,.122,
     2,.116, 2.48,.1, 3,.092, 3.9,.082, 40,.03,
     - 4.,
     0,.024, .2,.024, .6,.026, .8,.028, .9,.035, 1.3,.093, 1.5,.122,
     2,.116, 2.48,.1, 3,.092, 3.9,.082, 40,.03,
      0.,
     0,.026, .2,.026, .6,.026, .8,.024, .9,.036, 1.3,.092, 1.5,.118,
     2,.106, 2.48,.091, 3,.082, 3.9,.074, 40,.022,
      5.,
     0,.042, .2,.042, .6,.04, .8,.042, .9,.076, 1.3,.124, 1.5,.142,
     2,.124, 2.48,.098, 3,.088, 3.9,.079, 40,.033,
      10.,
     0,.076, .2,.076, .6,.08, .8,.1, .9,.13, 1.3,.194, 1.5,.192,
     2,.165, 2.48,.127, 3,.114, 3.9,.095, 40,.057,
      20.,
     0,.36, .2,.36, .6,.362, .8,.44, .9,.41, 1.3,.39, 1.5,.36 , 2,.32,
     2.48,.242, 3,.224, 3.9,.216, 40,.238,
      30.,
     0,.36, .2,.36, .6,.36, .8,.44, .9,.41, 1.3,.39, 1.5,.36, 2,.32,
     2.48,.44, 3,.418, 3.9,.4, 40,.3,
     $
```

l$TAB
TABLE   = 6hclt   ,2,6hmach ,6halpha ,12,7,1,1,1,1,1,1,1,1,1,
-20.,
0,-.07, .2,-.08, .6,-.12, .8,-.12, .9,-.12, 1.3,-.12, 1.5,-.12,
2,-.13, 2.48,-.14, 3,-.12, 3.9,-.1, 40,-.14,
 4.,
0,-.07, .2,-.08, .6,-.12, .8,-.12, .9,-.12, 1.3,-.12, 1.5,-.12,
2,-.13, 2.48,-.14, 3,-.12, 3.9,-.1, 40,-.14,
 0.,
0,.08, .2,.08, .6,.08, .8,.06, .9,.06, 1.3,.07, 1.5,.04, 2,0.0,
2.48,-.02, 3,-.03, 3.9,-.04, 40,.03,
 5.,
0,.29, .2,.29, .6,.29, .8,.28, .9,.28, 1.3,.3, 1.5,.24, 2,.17,
2.48,.12, 3,.09, 3.9,.08, 40,.21,
 10.,
0,.5, .2,.6, .6,.49, .8,.48, .9,.52, 1.3,.52, 1.5,.41, 2,.33,
2.48,.25, 3,.2, 3.9,.15, 40,.4,
 20.,
0,.94, .2,.94, .6,.92, .8,.9, .9,.94, 1.3,.89, 1.5,.75, 2,.63,
2.48,.51, 3,.43, 3.9,.39, 40,.76,
 30.,
0,.94, .2,.94, .6,.92, .8,.9, .9,.94, 1.3,.89, 1.5,.75, 2,.68,
2.48,.67, 3,.65, 3.9,.62, 40,.76,
 $
l$TAB
TABLE   = 6hcmat ,0,0.0,
 $
l$TAB
TABLE   = 6hxcgt ,1,6hweicon,5,1,1,1,
0,87.64, 202250,93.93, 404500,99.68, 606750,104.04, 809000,104.37,
 $
l$TAB
TABLE   = 6hycgt ,0,0.0,
 $
l$TAB
TABLE   = 6hzcgt ,1,6hweicon,5,1,1,1,
0,31.33, 202250,31.5, 404500,31.75, 606750,32.42, 809000,33.83,
ENDPHS =   1,
 $
l$GENDAT
EVENT   =   10,
CRITR   = 6htdurp ,
VALUE   = 100.0,

108

```
ENDPHS    =       1,
$
1$GENDAT
EVENT    =       11,
CRITR    =       6htdurp ,
VALUE    =       150.0,
NPC(1)   =       2,              / Make conic calculations and print
ENDPHS   =       1,
$
1$GENDAT
EVENT    =       12,
CRITR    =       6hwprop ,
VALUE    =       0.,
ENDPHS   =       1,
ENDPRB   =       1,
ENDJOB   =       1,
$
```

APPENDIX E. POST TABLES USED IN G-POST

This appendix contains the tables used to generate the G-POST file in Appendix D.

vacuum.tab
table   = 6htvc1t ,0,5472000.0,

exitarea.tab
table   = 6hae1t ,0,232.5,

dragforce.tab
table   = 6hcdt   ,2,6hmach ,6halpha ,12,5,1,1,1,1,1,1,1,1,
-20.,
0.0, 1.456, .5, 1.585, .7, 1.598, .8, 1.242, 1., 3.157, 1.2,2.996,
1.5, 1.816, 2.0, 1.301, 3., .850, 5., .482, 7., .382, 10., .396,
- 5.,
0.0, .263, .5, .338, .7, .110, .8, .302, 1., .690, 1.2, .671,
1.5, .563, 2.0, .480, 3., .383, 5., .256, 7., .212, 10., .210,
 0.,
0.0, .180, .5, .18 , .7, .200, .8, .251, 1., .495, 1.2, .502,
1.5, .485, 2.0, .456, 3., .391, 5., .272, 7., .231, 10., .231,
 5.,
0.0, .263, .5, .338, .7, .110, .8, .302, 1., .690, 1.2, .671,
1.5, .563, 2.0, .480, 3., .383, 5., .256, 7., .212, 10., .210,
 20.,
0.0, 1.456, .5, 1.585, .7, 1.598, .8, 1.242, 1., 3.157, 1.2,2.996,
1.5, 1.816, 2.0, 1.301, 3., .850, 5., .482, 7., .382, 10., .396,

liftforce.tab
table   = 6hclt   ,2,6hmach ,6halpha ,12,4,1,1,1,1,1,1,1,1,
-20.,
0.0, -1.010, .5,-1.025, .7, -.99, .8, -.815, 1.,-1.08 , 1.2,-1.11,
1.5, -.895, 2.0, -.788, 3.,-.635, 5., -.480, 7., -.43 , 10., -.43,
 0.,
0.0, .015, .5, .04 , .7, .01 , .8, -.045, 1., .08 , 1.2, .038,
1.5 -.02 , 2.0, -.108, 3.,-.145, 5., -.15, 7., -.15 , 10.,-.15 ,
 5.,
0.0, .545, .5, .75, .7, .53, .8, .365, 1., .69 , 1.2, .638,
1.5, .43 , 2. , .242, 3., .11, 5., .025 , 7., .00 , 10., .00 ,
 20.,
0.0, 2.135, .5, 2.24 , .7, 2.09, .8,1.595 , 1., 2.52 , 1.2,2.438,
1.5, 1.78 , 2. , 1.292, 3., .875, 5., .55 , 7., .45 , 10., .45 ,

momentcoeff.tab
```
table   = 6hcmat ,1,6hmach ,12,1,1,1,
0.0, .019, .7, .0218, .9, .0302, 1., .023, 1.2,-.011, 1.5,-.032,
1.8,-.0395, 2.,-.0419, 3.,-.0396, 5.,-.0187, 7.,-.0082, 10., 0.0,
```

aeroref.tab
```
table   = 6hxreft ,1,6hmach ,12,1,1,1,
0.0,137.86, .7,140.05, .9,136.77, 1.,147.71, 1.2,145.52,
1.5,144.43, 1.8,141.58, 2.,138.3, 3.,131.74, 5.,118.83,
7.,109.42, 10.,91.91,
```

vacuum2.tab
```
table   = 6htvc1t ,0,1431000.0,
```

exitarea2.tab
```
table   = 6hae1t ,0,154.54,
```

dragforce2.tab
```
table   = 6hcdt ,2,6hmach ,6halpha ,12,7,1,1,1,1,1,1,1,1,
-20.,
0,.024, .2,.024, .6,.026, .8,.028, .9,.035, 1.3,.093, 1.5,.122,
2,.116, 2.48,.1, 3,.092, 3.9,.082, 40,.03,
- 4.,
0,.024, .2,.024, .6,.026, .8,.028, .9,.035, 1.3,.093, 1.5,.122,
2,.116, 2.48,.1, 3,.092, 3.9,.082, 40,.03,
 0.,
0,.026, .2,.026, .6,.026, .8,.024, .9,.036, 1.3,.092, 1.5,.118,
2,.106, 2.48,.091, 3,.082, 3.9,.074, 40,.022,
 5.,
0,.042, .2,.042, .6,.04, .8,.042, .9,.076, 1.3,.124, 1.5,.142,
2,.124, 2.48,.098, 3,.088, 3.9,.079, 40,.033,
 10.,
0,.076, .2,.076, .6,.08, .8,.1, .9,.13, 1.3,.194, 1.5,.192,
2,.165, 2.48,.127, 3,.114, 3.9,.095, 40,.057,
 20.,
0,.36, .2,.36, .6,.362, .8,.44, .9,.41, 1.3,.39, 1.5,.36 , 2,.32,
2.48,.242, 3,.224, 3.9,.216, 40,.238,
 30.,
0,.36, .2,.36, .6,.36, .8,.44, .9,.41, 1.3,.39, 1.5,.36, 2,.32,
2.48,.44, 3,.418, 3.9,.4, 40,.3,
```

liftforce2.tab

table = 6hclt ,2,6hmach ,6halpha ,12,7,1,1,1,1,1,1,1,1,
-20.,
0,-.07, .2,-.08, .6,-.12, .8,-.12, .9,-.12, 1.3,-.12, 1.5,-.12,
2,-.13, 2.48,-.14, 3,-.12, 3.9,-.1, 40,-.14,
  4.,
0,-.07, .2,-.08, .6,-.12, .8,-.12, .9,-.12, 1.3,-.12, 1.5,-.12,
2,-.13, 2.48,-.14, 3,-.12, 3.9,-.1, 40,-.14,
  0.,
0,.08, .2,.08, .6,.08, .8,.06, .9,.06, 1.3,.07, 1.5,.04, 2,0.0,
2.48,-.02, 3,-.03, 3.9,-.04, 40,.03,
  5.,
0,.29, .2,.29, .6,.29, .8,.28, .9,.28, 1.3,.3, 1.5,.24, 2,.17,
2.48,.12, 3,.09, 3.9,.08, 40,.21,
 10.,
0,.5, .2,.6, .6,.49, .8,.48, .9,.52, 1.3,.52, 1.5,.41, 2,.33,
2.48,.25, 3,.2, 3.9,.15, 40,.4,
 20.,
0,.94, .2,.94, .6,.92, .8,.9, .9,.94, 1.3,.89, 1.5,.75, 2,.63,
2.48,.51, 3,.43, 3.9,.39, 40,.76,
 30.,
0,.94, .2,.94, .6,.92, .8,.9, .9,.94, 1.3,.89, 1.5,.75, 2,.68,
2.48,.67, 3,.65, 3.9,.62, 40,.76,

momentcoeff2.tab

table   = 6hcmat ,0,0.0,

xcentergrav.tab

table   = 6hxcgt ,1,6hweicon,5,1,1,1,
0,87.64, 202250,93.93, 404500,99.68, 606750,104.04, 809000,104.37,

ycentergrav.tab

table   = 6hycgt ,0,0.0,

zcentergrav.tab

table   = 6hzcgt ,1,6hweicon,5,1,1,1,
0,31.33, 202250,31.5, 404500,31.75, 606750,32.42, 809000,33.83,

LIST OF REFERENCES

1. Brauer, G., Cornick, D., Olson, D., Petersen, F., Stevenson, R., Utilization Manual, *Final Report for the Program to Optimize Simulated Trajectories (POST) Volume II*, Martin Marietta Corporation, Denver, Colorado, September, 1989.

2. Integrated Computer Solutions Incorporated (ICS), <u>The Builder Xcessory User's Guide Unix Edition</u>, ICS, Cambridge, Massachusetts, 1994.

3. McMinds, D., <u>Mastering OSF/Motif Widgets 2nd Edition</u>, Addison-Wesley Publishing Company, Reading, Massachusetts, 1993.

4. Nicholson, J., *A Primer for Post*, AE 4900 Course Paper, Department of Aeronautics, Naval Postgraduate School, Summer-Fall 1993.

5. The MathWorks Inc., <u>Matlab User's Guide</u>, The MathWorks Inc., South Natick, Massachusetts, April 1989.

6. Zyda, M., "Rotate3.C", *Course Notes for CS4202*, Department of Computer Science, Naval Postgraduate School, March 1994.

INITIAL DISTRIBUTION LIST